Namespace AuthorHaikuApi.Tests.UnitTests

Classes

AuthorHaikuServiceTests

Class AuthorHaikuServiceTests

Namespace: AuthorHaikuApi.Tests.UnitTests

Assembly: HaikuApi.Tests.dll

```
public class AuthorHaikuServiceTests
```

Inheritance

object ← AuthorHaikuServiceTests

Inherited Members

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

Constructors

AuthorHaikuServiceTests(ITestOutputHelper)

```
public AuthorHaikuServiceTests(ITestOutputHelper output)
```

Parameters

`output` ITestOutputHelper

Methods

CreateAuthorHaikuAsync_AddAuthorHaiku()

```
[Fact]
public Task CreateAuthorHaikuAsync_AddAuthorHaiku()
```

Returns

Task

DeleteAuthorHaikuAsync_DeleteAuthorHaiku()

```
[Fact]
public Task DeleteAuthorHaikuAsync_DeleteAuthorHaiku()
```

Returns

Task

GetAuthorHaikuAsync_ReturnAuthorHaiku()

```
[Fact]
public Task GetAuthorHaikuAsync_ReturnAuthorHaiku()
```

Returns

Task

UpdateAuthorHaikuAsync_UpdateAuthorHaiku()

```
[Fact]
public Task UpdateAuthorHaikuAsync_UpdateAuthorHaiku()
```

Returns

Task

Namespace Haiku.API.Controllers

Classes

AuthorController

AuthorHaikuController

LoginController

RegisterController

UserController

UserHaikuController

UserProfileController

Class AuthorController

Namespace: Haiku.API.Controllers

Assembly: Haiku.API.dll

```
[Route("api/[controller]")]
[ApiController]
public class AuthorController : ControllerBase
```

Inheritance

object ← ControllerBase ← AuthorController

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.ValidationProblem(ValidationProblemDetails) ,
ControllerBase.ValidationProblem(ModelStateDictionary) , ControllerBase.ValidationProblem() ,
ControllerBase.ValidationProblem(string, string, int?, string, string, ModelStateDictionary) ,
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

AuthorController(IAuthorService, IPaginationService, IXmlSerializationService, ILogger<AuthorController>)

```
public AuthorController(IAuthorService authorService, IPaginationService paginationService,
IXmlSerializationService xmlSerializationService, ILogger<AuthorController> logger)
```

Parameters

`authorService` IAuthorService

`paginationService` IPaginationService

`xmlSerializationService` IXmlSerializationService

`logger` ILogger<AuthorController>

Methods

DeleteAuthorAsync(long)

Deletes an author specified by their ID.

```
[HttpDelete("{authorId}")]
[Produces("application/xml", new string[] { })]
[Authorize(Policy = "AdminRole")]
public Task<IActionResult> DeleteAuthorAsync(long authorId)
```

Parameters

`authorId` long

The ID of the author to delete.

Returns

Task<IActionResult>

An IActionResult indicating the outcome of the delete operation. Returns 204 No Content on a successful deletion, or 404 Not Found if the author does not exist.

GetAllPaginatedAuthorsAsync(int, int, string?)

Retrieves a paginated list of authors.

```
[HttpGet]
[Produces("application/xml", new string[] { })]
public Task<IActionResult> GetAllPaginatedAuthorsAsync(int currentPage = 1, int pageSize =
10, string? searchOption = null)
```

Parameters

`currentPage` int

The current page number (default is `1`).

`pageSize` int

The number of authors per page (default is `10`).

`searchOption` string

An optional search term for filtering authors. If null, no filtering is applied.

Returns

Task<IActionResult>

A IActionResult containing a list of authors along with pagination metadata.

GetAuthorByIdAsync(long)

Retrieves a specific author by their unique identifier.

```
[HttpGet("{authorId}", Name = "AuthorDetails")]
[Produces("application/xml", new string[] { })]
public Task<ActionResult<AuthorDto>> GetAuthorByIdAsync(long authorId)
```

Parameters

`authorId` long

The unique identifier of the author to retrieve.

Returns

Task<ActionResult<AuthorDto>>

An ActionResult<TValue> containing the author details if found, or a 404 Not Found status if the author does not exist.

PostAuthorAsync(AuthorDto)

Creates a new author using the provided author details.

```
[HttpPost]
[Consumes("application/xml", new string[] { })]
[Produces("application/xml", new string[] { })]
[Authorize(Policy = "AdminRole")]
public Task<ActionResult<AuthorDto>> PostAuthorAsync(AuthorDto authorDto)
```

Parameters

`authorDto` AuthorDto

The details of the author to create.

Returns

Task<ActionResult<AuthorDto>>

An ActionResult<TValue> representing the created author details, or a 400 Bad Request if the input validation fails.

PutAuthorAsync(long, AuthorDto)

Updates the details of an existing author specified by their ID.

```
[HttpPut("{authorId}")]
[Consumes("application/xml", new string[] { })]
[Produces("application/xml", new string[] { })]
[Authorize(Policy = "AdminRole")]
public Task<IActionResult> PutAuthorAsync(long authorId, AuthorDto authorDto)
```

Parameters

`authorId` long

The ID of the author to update.

`authorDto` AuthorDto

The updated details of the author.

Returns

Task<IActionResult>

An IActionResult indicating the outcome of the update operation. Returns 204 No Content on a successful update, or 400 Bad Request if the input validation fails, or 404 Not Found if the author does not exist.

Class AuthorHaikuController

Namespace: Haiku.API.Controllers

Assembly: Haiku.API.dll

```
[Route("api/[controller]")]
[ApiController]
public class AuthorHaikuController : ControllerBase
```

Inheritance

object ← ControllerBase ← AuthorHaikuController

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.ValidationProblem(ValidationProblemDetails) ,
ControllerBase.ValidationProblem(ModelStateDictionary) , ControllerBase.ValidationProblem() ,
ControllerBase.ValidationProblem(string, string, int?, string, string, ModelStateDictionary) ,
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

AuthorHaikuController(IAuthorHaikuService, IPaginationService, IXmlSerializationService, ILogger<AuthorHaikuController>)

```
public AuthorHaikuController(IAuthorHaikuService authorHaikuService, IPaginationService
paginationService, IXmlSerializationService xmlSerializationService,
ILogger<AuthorHaikuController> logger)
```

Parameters

`authorHaikuService` IAuthorHaikuService

`paginationService` IPaginationService

`xmlSerializationService` IXmlSerializationService

`logger` ILogger<AuthorHaikuController>

Methods

DeleteAuthorHaikuAsync(long)

Deletes an author haiku specified by their ID.

```
[HttpDelete("{authorHaikuId}")]
[Produces("application/xml", new string[] { })]
[Authorize(Policy = "AdminRole")]
public Task<IActionResult> DeleteAuthorHaikuAsync(long authorHaikuId)
```

Parameters

`authorHaikuId` long

The ID of the author haiku to delete.

Returns

Task<IActionResult>

An IActionResult indicating the outcome of the delete operation. Returns 204 No Content on a successful deletion, or 404 Not Found if the author haiku does not exist.

GetAllPaginatedAuthorHaikusAsync(int, int, string?)

Retrieves a paginated list of author haikus.

```
[HttpGet]
[Produces("application/xml", new string[] { })]
public Task<IActionResult> GetAllPaginatedAuthorHaikusAsync(int currentPage = 1, int
pageSize = 10, string? searchOption = null)
```

Parameters

`currentPage` int

The current page number (default is `1`).

`pageSize` int

The number of author haikus per page (default is `10`).

`searchOption` string

An optional search term for filtering author haikus. If null, no filtering is applied.

Returns

Task<IActionResult>

A IActionResult containing a list of author haikus along with pagination metadata.

GetAuthorHaikuByIdAsync(long)

Retrieves a specific author haiku by their unique identifier.

```
[HttpGet("{authorHaikuId}", Name = "AuthorHaikuDetails")]
[Produces("application/xml", new string[] { })]
public Task<ActionResult<AuthorHaikuDto>> GetAuthorHaikuByIdAsync(long authorHaikuId)
```

Parameters

`authorHaikuId` long

The unique identifier of the author haiku to retrieve.

Returns

Task<ActionResult<AuthorHaikuDto>>

An ActionResult<TValue> containing the author haiku details if found, or a 404 Not Found status if the author haiku does not exist.

GetPaginatedAuthorHaikusByAuthorIdAsync(long, int, int, string?)

Retrieves a paginated list of author haikus from an author.

```
[HttpGet("Author/{authorId}")]
[Produces("application/xml", new string[] { })]
```

```
public Task<IActionResult> GetPaginatedAuthorHaikusByAuthorIdAsync(long authorId, int
currentPage = 1, int pageSize = 10, string? searchOption = null)
```

Parameters

`authorId` long

The author identifier to retrieve with.

`currentPage` int

The current page number (default is `1`).

`pageSize` int

The number of author haikus per page (default is `10`).

`searchOption` string

An optional search term for filtering author haikus. If null, no filtering is applied.

Returns

Task<IActionResult>

A IActionResult containing a list of author haikus along with pagination metadata.

PostAuthorHaikuAsync(AuthorHaikuDto)

Creates a new author haiku using the provided author haiku details.

```
[HttpPost]
[Consumes("application/xml", new string[] { })]
[Produces("application/xml", new string[] { })]
[Authorize(Policy = "AdminRole")]
public Task<ActionResult<AuthorHaikuDto>> PostAuthorHaikuAsync(AuthorHaikuDto
authorHaikuDto)
```

Parameters

`authorHaikuDto` AuthorHaikuDto

The details of the author haiku to create.

Returns

Task<ActionResult<AuthorHaikuDto>>

An ActionResult<TValue> representing the created author haiku details, or a 400 Bad Request if the input validation fails.

PutAuthorHaikuAsync(long, AuthorHaikuDto)

Updates the details of an existing author haiku specified by their ID.

```
[HttpPut("{authorHaikuId}")]
[Consumes("application/xml", new string[] { })]
[Produces("application/xml", new string[] { })]
[Authorize(Policy = "AdminRole")]
public Task<IActionResult> PutAuthorHaikuAsync(long authorHaikuId,
AuthorHaikuDto authorHaikuDto)
```

Parameters

`authorHaikuId` long

The ID of the author haiku to update.

`authorHaikuDto` AuthorHaikuDto

The updated details of the author haiku.

Returns

Task<IActionResult>

An IActionResult indicating the outcome of the update operation. Returns 204 No Content on a successful update, or 400 Bad Request if the input validation fails, or 404 Not Found if the author haiku does not exist.

Class LoginController

Namespace: Haiku.API.Controllers

Assembly: Haiku.API.dll

```
[Route("api/[controller]")]
[ApiController]
public class LoginController : ControllerBase
```

Inheritance

object ← ControllerBase ← LoginController

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.ValidationProblem(ValidationProblemDetails) ,
ControllerBase.ValidationProblem(ModelStateDictionary) , ControllerBase.ValidationProblem() ,
ControllerBase.ValidationProblem(string, string, int?, string, string, ModelStateDictionary) ,
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

LoginController(IAuthService, ILogger<LoginController>)

```
public LoginController(IAuthService authService, ILogger<LoginController> logger)
```

Parameters

`authService` IAuthService

`logger` ILogger<LoginController>

Methods

PostLoginUser(LoginDto)

Authenticates a user based on the provided login credentials.

```
[HttpPost]
[Consumes("application/xml", new string[] { })]
[Produces("application/xml", new string[] { })]
public Task<IActionResult> PostLoginUser(LoginDto loginDto)
```

Parameters

`loginDto` LoginDto

The login details including username and password.

Returns

Task<IActionResult>

An IActionResult containing the authentication token on successful login, or a 400 Bad Request if the input validation fails, or a 401 Unauthorized if authentication fails.

Class RegisterController

Namespace: Haiku.API.Controllers

Assembly: Haiku.API.dll

```
[Route("api/[controller]")]
[ApiController]
public class RegisterController : ControllerBase
```

Inheritance

object ← ControllerBase ← RegisterController

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.ValidationProblem(ValidationProblemDetails) ,
ControllerBase.ValidationProblem(ModelStateDictionary) , ControllerBase.ValidationProblem() ,
ControllerBase.ValidationProblem(string, string, int?, string, string, ModelStateDictionary) ,
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

RegisterController(IUserService, ILogger<RegisterController>)

```
public RegisterController(IUserService userService, ILogger<RegisterController> logger)
```

Parameters

`userService` IUserService

`logger` ILogger<RegisterController>

Methods

GetExistingUsernameVerification(string)

Verifies if a given username already exists in the system.

```
[HttpGet("verify-username/{username}")]
[Produces("application/xml", new string[] { })]
public Task<IActionResult> GetExistingUsernameVerification(string username)
```

Parameters

`username` string

The username to verify for existence.

Returns

Task<IActionResult>

An IActionResult containing a boolean indicating whether the username exists, or a 400 Bad Request if the username is null or empty.

PostRegisterUser(RegisterDto)

Registers a new user in the system.

```
[HttpPost]
[Consumes("application/xml", new string[] { })]
[Produces("application/xml", new string[] { })]
public Task<IActionResult> PostRegisterUser(RegisterDto registerUserDto)
```

Parameters

`registerUserDto` RegisterDto

The details of the user to register.

Returns

Task<IActionResult>

An IActionResult containing the newly registered user details, or a 400 Bad Request if the input validation fails.

Class UserController

Namespace: Haiku.API.Controllers

Assembly: Haiku.API.dll

```
[Route("api/[controller]")]
[ApiController]
public class UserController : ControllerBase
```

Inheritance

object ← ControllerBase ← UserController

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.ValidationProblem(ValidationProblemDetails) ,
ControllerBase.ValidationProblem(ModelStateDictionary) , ControllerBase.ValidationProblem() ,
ControllerBase.ValidationProblem(string, string, int?, string, string, ModelStateDictionary) ,
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

UserController(IUserService, IPaginationService, IXmlSerializationService, ILogger<UserController>)

```
public UserController(IUserService userService, IPaginationService paginationService,
IXmlSerializationService xmlSerializationService, ILogger<UserController> logger)
```

Parameters

`userService` IUserService

`paginationService` IPaginationService

`xmlSerializationService` IXmlSerializationService

`logger` ILogger<UserController>

Methods

DeleteUserAsync(long)

Deletes an user specified by their ID.

```
[HttpDelete("{userId}")]
[Produces("application/xml", new string[] { })]
[Authorize]
public Task<IActionResult> DeleteUserAsync(long userId)
```

Parameters

`userId` long

The ID of the user to delete.

Returns

Task<IActionResult>

An IActionResult indicating the outcome of the delete operation. Returns 204 No Content on a successful deletion, or 404 Not Found if the user does not exist.

GetAllPaginatedUsersAsync(int, int, string?)

Retrieves a paginated list of users.

```
[HttpGet]
[Produces("application/xml", new string[] { })]
public Task<IActionResult> GetAllPaginatedUsersAsync(int currentPage = 1, int pageSize = 10,
string? searchOption = null)
```

Parameters

`currentPage` int

The current page number (default is `1`).

`pageSize` int

The number of users per page (default is `10`).

`searchOption` string

An optional search term for filtering users. If null, no filtering is applied.

Returns

Task<IActionResult>

A IActionResult containing a list of users along with pagination metadata.

GetUserByIdAsync(long)

Retrieves a specific user by their unique identifier.

```
[HttpGet("{userId}", Name = "UserDetails")]
[Produces("application/xml", new string[] { })]
public Task<ActionResult<UserDto>> GetUserByIdAsync(long userId)
```

Parameters

`userId` long

The unique identifier of the user to retrieve.

Returns

Task<ActionResult<UserDto>>

An ActionResult<TValue> containing the user details if found, or a 404 Not Found status if the user does not exist.

PutUserAsync(long, UserDto)

Updates the details of an existing user specified by their ID.

```
[HttpPut("{userId}")]
[Consumes("application/xml", new string[] { })]
[Produces("application/xml", new string[] { })]
[Authorize]
public Task<IActionResult> PutUserAsync(long userId, UserDto userDto)
```

Parameters

`userId` long

The ID of the user to update.

`userDto` UserDto

The updated details of the user.

Returns

Task<IActionResult>

An IActionResult indicating the outcome of the update operation. Returns 204 No Content on a successful update, or 400 Bad Request if the input validation fails, or 404 Not Found if the user does not exist.

Class UserHaikuController

Namespace: Haiku.API.Controllers

Assembly: Haiku.API.dll

```
[Route("api/[controller]")]
[ApiController]
public class UserHaikuController : ControllerBase
```

Inheritance

object ← ControllerBase ← UserHaikuController

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.ValidationProblem(ValidationProblemDetails) ,
ControllerBase.ValidationProblem(ModelStateDictionary) , ControllerBase.ValidationProblem() ,
ControllerBase.ValidationProblem(string, string, int?, string, string, ModelStateDictionary) ,
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

UserHaikuController(IUserHaikuService, IPaginationService, IXmlSerializationService, ILogger<UserHaikuController>)

```
public UserHaikuController(IUserHaikuService userHaikuService, IPaginationService
paginationService, IXmlSerializationService xmlSerializationService,
ILogger<UserHaikuController> logger)
```

Parameters

`userHaikuService` IUserHaikuService

`paginationService` IPaginationService

`xmlSerializationService` IXmlSerializationService

`logger` ILogger<UserHaikuController>

Methods

DeleteUserHaikuAsync(long)

Deletes an user haiku specified by their ID.

```
[HttpDelete("{userHaikuId}")]
[Produces("application/xml", new string[] { })]
[Authorize]
public Task<IActionResult> DeleteUserHaikuAsync(long userHaikuId)
```

Parameters

`userHaikuId` long

The ID of the user haiku to delete.

Returns

Task<IActionResult>

An IActionResult indicating the outcome of the delete operation. Returns 204 No Content on a successful deletion, or 404 Not Found if the user haiku does not exist.

GetAllPaginatedHaikusAsync(int, int, string?)

Retrieves a paginated list of user haikus.

```
[HttpGet]
[Produces("application/xml", new string[] { })]
public Task<IActionResult> GetAllPaginatedHaikusAsync(int currentPage = 1, int pageSize =
10, string? searchOption = null)
```

Parameters

`currentPage` int

The current page number (default is `1`).

`pageSize` int

The number of user haikus per page (default is `10`).

`searchOption` string

An optional search term for filtering user haikus. If null, no filtering is applied.

Returns

Task<IActionResult>

A IActionResult containing a list of user haikus along with pagination metadata.

GetPaginatedUserHaikusByUserIdAsync(long, int, int, string?)

Retrieves a paginated list of user haikus from an user.

```
[HttpGet("User/{userId}")]
[Produces("application/xml", new string[] { })]
public Task<IActionResult> GetPaginatedUserHaikusByUserIdAsync(long userId, int currentPage
= 1, int pageSize = 10, string? searchOption = null)
```

Parameters

`userId` long

The user identifier to retrieve with.

`currentPage` int

The current page number (default is `1`).

`pageSize` int

The number of user haikus per page (default is `10`).

`searchOption` string

An optional search term for filtering user haikus. If null, no filtering is applied.

Returns

Task<IActionResult>

A IActionResult containing a list of user haikus along with pagination metadata.

GetUserHaikuByIdAsync(long)

Retrieves a specific user haiku by their unique identifier.

```
[HttpGet("{userHaikuId}", Name = "UserHaikuDetails")]
[Produces("application/xml", new string[] { })]
public Task<ActionResult<UserHaikuDto>> GetUserHaikuByIdAsync(long userHaikuId)
```

Parameters

`userHaikuId` long

The unique identifier of the user haiku to retrieve.

Returns

Task<ActionResult<UserHaikuDto>>

An ActionResult<TValue> containing the user haiku details if found, or a 404 Not Found status if the user haiku does not exist.

PostUserHaikuAsync(UserHaikuDto)

Creates a new user haiku using the provided user haiku details.

```
[HttpPost]
[Consumes("application/xml", new string[] { })]
[Produces("application/xml", new string[] { })]
[Authorize]
public Task<ActionResult<UserHaikuDto>> PostUserHaikuAsync(UserHaikuDto userHaikuDto)
```

Parameters

`userHaikuDto` UserHaikuDto

The details of the user haiku to create.

Returns

Task<ActionResult<UserHaikuDto>>

An ActionResult<TValue> representing the created user haiku details, or a 400 Bad Request if the input validation fails.

PutUserHaikuAsync(long, UserHaikuDto)

Updates the details of an existing user haiku specified by their ID.

```
[HttpPut("{userHaikuId}")]
[Consumes("application/xml", new string[] { })]
[Produces("application/xml", new string[] { })]
[Authorize]
public Task<IActionResult> PutUserHaikuAsync(long userHaikuId, UserHaikuDto userHaikuDto)
```

Parameters

`userHaikuId` long

The ID of the user haiku to update.

`userHaikuDto` UserHaikuDto

The updated details of the user haiku.

Returns

Task<IActionResult>

An IActionResult indicating the outcome of the update operation. Returns 204 No Content on a successful update, or 400 Bad Request if the input validation fails, or 404 Not Found if the user haiku does not exist.

Class UserProfileController

Namespace: Haiku.API.Controllers

Assembly: Haiku.API.dll

```
[Route("api/[controller]")]
[ApiController]
public class UserProfileController : ControllerBase
```

Inheritance

object ← ControllerBase ← UserProfileController

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.ValidationProblem(ValidationProblemDetails) ,
ControllerBase.ValidationProblem(ModelStateDictionary) , ControllerBase.ValidationProblem() ,
ControllerBase.ValidationProblem(string, string, int?, string, string, ModelStateDictionary) ,
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

UserProfileController(IUserProfileService, IPaginationService, ILogger<UserProfileController>)

```
public UserProfileController(IUserProfileService userProfileService, IPaginationService
paginationService, ILogger<UserProfileController> logger)
```

Parameters

`userProfileService` IUserProfileService

`paginationService` IPaginationService

`logger` ILogger<UserProfileController>

Methods

GetAllUserProfilesByUserIdsAsync(List<long>)

Retrieves user profiles for the specified user IDs.

```
[HttpGet("profiles-by-ids")]
[Produces("application/xml", new string[] { })]
public Task<IActionResult> GetAllUserProfilesByUserIdsAsync(List<long> userIds)
```

Parameters

`userIds` List<long>

A list of user IDs for which to fetch profiles.

Returns

Task<IActionResult>

An IActionResult containing the list of user profiles, or a 500 Internal Server Error if an unexpected error occurs.

GetUserProfileByIdAsync(long)

Retrieves a specific user profile by their unique identifier.

```
[HttpGet("{userProfileId}", Name = "UserProfileDetails")]
[Produces("application/xml", new string[] { })]
public Task<ActionResult<UserProfileDto>> GetUserProfileByIdAsync(long userProfileId)
```

Parameters

`userProfileId` long

The unique identifier of the user profile to retrieve.

Returns

Task<ActionResult<UserProfileDto>>

An ActionResult<TValue> containing the user profile details if found, or a 404 Not Found status if the user profile does not exist.

GetUserProfileByUserIdAsync(long)

Retrieves a specific user profile by their user's unique identifier.

```
[HttpGet("user/{userId}")]
[Produces("application/xml", new string[] { })]
public Task<ActionResult<UserProfileDto>> GetUserProfileByUserIdAsync(long userId)
```

Parameters

`userId` long

The unique identifier of the user profile user to retrieve.

Returns

Task<ActionResult<UserProfileDto>>

An ActionResult<TValue> containing the user profile details if found, or a 404 Not Found status if the user profile does not exist.

PostUserProfileAsync(UserProfileDto)

Creates a new user profile using the provided user profile details.

```
[HttpPost]
[Consumes("application/xml", new string[] { })]
[Produces("application/xml", new string[] { })]
[Authorize]
public Task<ActionResult<UserProfileDto>> PostUserProfileAsync(UserProfileDto
userProfileDto)
```

Parameters

`userProfileDto` UserProfileDto

Returns

Task<ActionResult<UserProfileDto>>

An ActionResult<TValue> representing the created user profile details, or a 400 Bad Request if the input validation fails.

PostUserProfileImage(IFormFile)

Uploads a profile image for the user.

```
[HttpPost("upload-profile-image")]
[Authorize]
public Task<IActionResult> PostUserProfileImage(IFormFile file)
```

Parameters

`file` IFormFile

The profile image file to be uploaded.

Returns

Task<IActionResult>

An IActionResult containing the name and file path of the uploaded image, or a 400 Bad Request if no file is uploaded.

PutUserProfileAsync(long, UserProfileDto)

Updates the details of an existing user profile specified by their ID.

```
[HttpPut("{userProfileId}")]
[Consumes("application/xml", new string[] { })]
[Authorize]
public Task<IActionResult> PutUserProfileAsync(long userProfileId,
UserProfileDto userProfileDto)
```

Parameters

`userProfileId` long

The ID of the user profile to update.

`userProfileDto` UserProfileDto

The updated details of the user profile .

Returns

Task<IActionResult>

An IActionResult indicating the outcome of the update operation. Returns 204 No Content on a successful update, or 400 Bad Request if the input validation fails, or 404 Not Found if the user profile does not exist.

Namespace Haiku.API.Database

Classes

HaikuAPIContext

Class HaikuAPIContext

Namespace: Haiku.API.Database

Assembly: Haiku.API.dll

```
public class HaikuAPIContext : DbContext, IInfrastructure<IServiceProvider>,
IDbContextDependencies, IDbSetCache, IDbContextPoolable, IResettableService,
IDisposable, IAsyncDisposable
```

Inheritance

object ← DbContext ← HaikuAPIContext

Implements

IInfrastructure<IServiceProvider>, IDbContextDependencies, IDbSetCache, IDbContextPoolable, IResettableService, IDisposable, IAsyncDisposable

Inherited Members

DbContext.Set<TEntity>() , DbContext.Set<TEntity>(string) ,
DbContext.OnConfiguring(DbContextOptionsBuilder) ,
DbContext.ConfigureConventions(ModelConfigurationBuilder) , DbContext.SaveChanges() ,
DbContext.SaveChanges(bool) , DbContext.SaveChangesAsync(CancellationToken) ,
DbContext.SaveChangesAsync(bool, CancellationToken) , DbContext.Dispose() ,
DbContext.DisposeAsync() , DbContext.Entry<TEntity>(TEntity) , DbContext.Entry(object) ,
DbContext.Add<TEntity>(TEntity) , DbContext.AddAsync<TEntity>(TEntity, CancellationToken) ,
DbContext.Attach<TEntity>(TEntity) , DbContext.Update<TEntity>(TEntity) ,
DbContext.Remove<TEntity>(TEntity) , DbContext.Add(object) ,
DbContext.AddAsync(object, CancellationToken) , DbContext.Attach(object) ,
DbContext.Update(object) , DbContext.Remove(object) , DbContext.AddRange(params object[]) ,
DbContext.AddRangeAsync(params object[]) , DbContext.AttachRange(params object[]) ,
DbContext.UpdateRange(params object[]) , DbContext.RemoveRange(params object[]) ,
DbContext.AddRange(IEnumerable<object>) ,
DbContext.AddRangeAsync(IEnumerable<object>, CancellationToken) ,
DbContext.AttachRange(IEnumerable<object>) , DbContext.UpdateRange(IEnumerable<object>) ,
DbContext.RemoveRange(IEnumerable<object>) , DbContext.Find(Type, params object[]) ,
DbContext.FindAsync(Type, params object[]) ,
DbContext.FindAsync(Type, object[], CancellationToken) , DbContext.Find<TEntity>(params object[]) ,
DbContext.FindAsync<TEntity>(params object[]) ,
DbContext.FindAsync<TEntity>(object[], CancellationToken) ,
DbContext.FromExpression<TResult>(Expression<Func<IQueryable<TResult>>>) ,

DbContext.Database , DbContext.ChangeTracker , DbContext.Model , DbContext.ContextId , DbContext.SavingChanges , DbContext.SavedChanges , DbContext.SaveChangesFailed , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

HaikuAPIContext()

```
public HaikuAPIContext()
```

HaikuAPIContext(DbContextOptions<HaikuAPIContext>)

```
public HaikuAPIContext(DbContextOptions<HaikuAPIContext> options)
```

Parameters

`options` DbContextOptions<HaikuAPIContext>

Properties

AuthorHaikus

```
public DbSet<AuthorHaiku> AuthorHaikus { get; set; }
```

Property Value

DbSet<AuthorHaiku>

Authors

```
public DbSet<Author> Authors { get; set; }
```

Property Value

DbSet<Author>

Roles

```
public DbSet<Role> Roles { get; set; }
```

Property Value

DbSet<Role>

UserHaikus

```
public DbSet<UserHaiku> UserHaikus { get; set; }
```

Property Value

DbSet<UserHaiku>

UserProfiles

```
public DbSet<UserProfile> UserProfiles { get; set; }
```

Property Value

DbSet<UserProfile>

Users

```
public DbSet<User> Users { get; set; }
```

Property Value

DbSet <User>

Methods

OnModelCreating(ModelBuilder)

Override this method to further configure the model that was discovered by convention from the entity types exposed in DbSet<TEntity> properties on your derived context. The resulting model may be cached and re-used for subsequent instances of your derived context.

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
```

Parameters

`modelBuilder` ModelBuilder

The builder being used to construct the model for this context. Databases (and other extensions) typically define extension methods on this object that allow you to configure aspects of the model that are specific to a given database.

Remarks

If a model is explicitly set on the options for this context (via UseModel(IModel)) then this method will not be run. However, it will still run when creating a compiled model.

See Modeling entity types and relationships for more information and examples.

Namespace Haiku.API.Dtos

Classes

AuthorDto

AuthorHaikuDto

JWTokenDto

LoginDto

PaginationMetaDataDto

RegisterDto

UserDto

UserHaikuDto

UserProfileDto

Class AuthorDto

Namespace: Haiku.API.Dtos

Assembly: Haiku.API.dll

```
public class AuthorDto
```

Inheritance

object ← AuthorDto

Inherited Members

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

Properties

Bio

```
[StringLength(300, ErrorMessage = "Bio length can't be more than 300 characters.")]
[MinLength(4, ErrorMessage = "Name length must be at least 4 characters.")]
public string? Bio { get; set; }
```

Property Value

string

Id

```
public long Id { get; set; }
```

Property Value

long

Name

```
[Required]
[StringLength(50, ErrorMessage = "Name length can't be more than 50 characters.")]
[MinLength(2, ErrorMessage = "Name length must be at least 2 characters.")]
public required string Name { get; set; }
```

Property Value

string

Class AuthorHaikuDto

Namespace: Haiku.API.Dtos

Assembly: Haiku.API.dll

```
public class AuthorHaikuDto
```

Inheritance

object ← AuthorHaikuDto

Inherited Members

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

Properties

AuthorId

```
[Required]
public long AuthorId { get; set; }
```

Property Value

long

Id

```
public long Id { get; set; }
```

Property Value

long

LineOne

```
[SyllableCount(5, ErrorMessage = "Must be five syllables")]
[Required]
[StringLength(50, ErrorMessage = "First line length can't be more than 50.")]
public required string LineOne { get; set; }
```

Property Value

string

LineThree

```
[SyllableCount(5, ErrorMessage = "Must be five syllables")]
[Required]
[StringLength(50, ErrorMessage = "Third line length can't be more than 50.")]
public required string LineThree { get; set; }
```

Property Value

string

LineTwo

```
[SyllableCount(7, ErrorMessage = "Must be seven syllables")]
[Required]
[StringLength(50, ErrorMessage = "Second line length can't be more than 50.")]
public required string LineTwo { get; set; }
```

Property Value

string

Title

```
[StringLength(50, ErrorMessage = "Title length can't be more than 50 characters.")]
[MinLength(2, ErrorMessage = "Title length must be at least 2 characters.")]
public string? Title { get; set; }
```

Property Value

string

Class JWTokenDto

Namespace: Haiku.API.Dtos

Assembly: Haiku.API.dll

```
public class JWTokenDto
```

Inheritance

object ← JWTokenDto

Inherited Members

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

Properties

Token

```
public required string Token { get; set; }
```

Property Value

string

Class LoginDto

Namespace: Haiku.API.Dtos

Assembly: Haiku.API.dll

```
public class LoginDto
```

Inheritance

object ← LoginDto

Inherited Members

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

Properties

Id

```
public long Id { get; set; }
```

Property Value

long

Password

```
[Required]
[StringLength(30, ErrorMessage = "Password length can't be more than 30 characters.")]
[MinLength(4, ErrorMessage = "Password length must be at least 8 characters.")]
public required string Password { get; set; }
```

Property Value

string

Username

```
[Required]
[StringLength(20, ErrorMessage = "Username length can't be more than 20 characters.")]
[MinLength(4, ErrorMessage = "Username length must be at least 4 characters.")]
public required string Username { get; set; }
```

Property Value

string

Class PaginationMetaDataDto

Namespace: Haiku.API.Dtos

Assembly: Haiku.API.dll

```
public class PaginationMetaDataDto
```

Inheritance

object ← PaginationMetaDataDto

Inherited Members

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

Properties

CurrentPage

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

Property Value

int

PageSize

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

Property Value

int

TotalCount

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

Property Value

int

TotalPages

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

Property Value

int

Class RegisterDto

Namespace: Haiku.API.Dtos

Assembly: Haiku.API.dll

```
public class RegisterDto
```

Inheritance

object ← RegisterDto

Inherited Members

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

Properties

ConfirmPassword

```
[Required]
[Compare("Password", ErrorMessage = "Passwords do not match.")]
public required string ConfirmPassword { get; set; }
```

Property Value

string

Id

```
public long Id { get; set; }
```

Property Value

long

Password

```
[Required]
[StringLength(30, ErrorMessage = "Password length can't be more than 30 characters.")]
[MinLength(8, ErrorMessage = "Password length must be at least 8 characters.")]
public required string Password { get; set; }
```

Property Value

string

Username

```
[Required]
[StringLength(20, ErrorMessage = "Username length can't be more than 20 characters.")]
[MinLength(4, ErrorMessage = "Username length must be at least 4 characters.")]
public required string Username { get; set; }
```

Property Value

string

Class UserDto

Namespace: Haiku.API.Dtos

Assembly: Haiku.API.dll

```
public class UserDto
```

Inheritance

object ← UserDto

Inherited Members

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

Properties

Id

```
public long Id { get; set; }
```

Property Value

long

Password

```
[Required]
[StringLengthIfNotEmpty(20, "Password")]
[MinLengthIfNotEmpty(8, "Password")]
public required string Password { get; set; }
```

Property Value

string

RoleId

```
[Required]
public long RoleId { get; set; }
```

Property Value

long

Username

```
[Required]
[StringLengthIfNotEmpty(20, "Username")]
[MinLengthIfNotEmpty(4, "Username")]
public required string Username { get; set; }
```

Property Value

string

Class UserHaikuDto

Namespace: Haiku.API.Dtos

Assembly: Haiku.API.dll

```
public class UserHaikuDto
```

Inheritance

object ← UserHaikuDto

Inherited Members

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

Properties

Id

```
public long Id { get; set; }
```

Property Value

long

LineOne

```
[SyllableCount(5, ErrorMessage = "Must be five syllables")]
[Required]
[StringLength(50, ErrorMessage = "First line length can't be more than 50.")]
public required string LineOne { get; set; }
```

Property Value

string

LineThree

```
[SyllableCount(5, ErrorMessage = "Must be five syllables")]
[Required]
[StringLength(50, ErrorMessage = "Third line length can't be more than 50.")]
public required string LineThree { get; set; }
```

Property Value

string

LineTwo

```
[SyllableCount(7, ErrorMessage = "Must be seven syllables")]
[Required]
[StringLength(50, ErrorMessage = "Second line length can't be more than 50.")]
public required string LineTwo { get; set; }
```

Property Value

string

Title

```
[StringLength(50, ErrorMessage = "Title length can't be more than 50 characters.")]
[MinLength(2, ErrorMessage = "Title length must be at least 2 characters.")]
public required string? Title { get; set; }
```

Property Value

string

UserId

```
[Required]
```

```
public long UserId { get; set; }
```

Property Value

long

Class UserProfileDto

Namespace: Haiku.API.Dtos

Assembly: Haiku.API.dll

```
public class UserProfileDto
```

Inheritance

object ← UserProfileDto

Inherited Members

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

Properties

Bio

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

Property Value

string

Id

```
public long Id { get; set; }
```

Property Value

long

ImageUrl

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

Property Value

string

UserId

```
[Required]
public long UserId { get; set; }
```

Property Value

long

Namespace Haiku.API.Exceptions

Classes

NotFoundException

NotSavedException

UsernameAlreadyTakenException

Class NotFoundException

Namespace: Haiku.API.Exceptions

Assembly: Haiku.API.dll

```
public class NotFoundException : Exception, ISerializable
```

Inheritance

object ← Exception ← NotFoundException

Implements

ISerializable

Inherited Members

Exception.GetBaseException() , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , Exception.SerializeObjectState , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Constructors

NotFoundException(string)

Initializes a new instance of the NotFoundException class with a specified error message.

```
public NotFoundException(string message)
```

Parameters

`message` string

The message that describes the error.

Class NotSavedException

Namespace: Haiku.API.Exceptions

Assembly: Haiku.API.dll

```
public class NotSavedException : Exception, ISerializable
```

Inheritance

object ← Exception ← NotSavedException

Implements

ISerializable

Inherited Members

Exception.GetBaseException() , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , Exception.SerializeObjectState , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Constructors

NotSavedException(string)

Initializes a new instance of the NotSavedException class with a specified error message.

```
public NotSavedException(string message)
```

Parameters

`message` string

The message that describes the error.

Class UsernameAlreadyTakenException

Namespace: Haiku.API.Exceptions

Assembly: Haiku.API.dll

```
public class UsernameAlreadyTakenException : Exception, ISerializable
```

Inheritance

object ← Exception ← UsernameAlreadyTakenException

Implements

ISerializable

Inherited Members

Exception.GetBaseException() , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , Exception.SerializeObjectState , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Constructors

UsernameAlreadyTakenException(string)

Initializes a new instance of the UsernameAlreadyTakenException class with a specified error message.

```
public UsernameAlreadyTakenException(string message)
```

Parameters

`message` string

The message that describes the error.

Namespace Haiku.API.Mapping

Classes

AuthorHaikuMapping

AuthorMapping

UserHaikuMapping

UserMapping

UserProfileMapping

Class AuthorHaikuMapping

Namespace: Haiku.API.Mapping

Assembly: Haiku.API.dll

```
public class AuthorHaikuMapping : Profile, IProfileExpression, IProfileConfiguration
```

Inheritance

object ← Profile ← AuthorHaikuMapping

Implements

IProfileExpression, IProfileConfiguration

Inherited Members

Profile.DisableConstructorMapping() , Profile.CreateProjection<TSource, TDestination>() , Profile.CreateProjection<TSource, TDestination>(MemberList) , Profile.CreateMap<TSource, TDestination>() , Profile.CreateMap<TSource, TDestination>(MemberList) , Profile.CreateMap(Type, Type) , Profile.CreateMap(Type, Type, MemberList) , Profile.ClearPrefixes() , Profile.ReplaceMemberName(string, string) , Profile.RecognizePrefixes(params string[]) , Profile.RecognizePostfixes(params string[]) , Profile.RecognizeDestinationPrefixes(params string[]) , Profile.RecognizeDestinationPostfixes(params string[]) , Profile.AddGlobalIgnore(string) , Profile.IncludeSourceExtensionMethods(Type) , Profile.ProfileName , Profile.AllowNullDestinationValues , Profile.AllowNullCollections , Profile.ShouldMapProperty , Profile.ShouldMapField , Profile.ShouldMapMethod , Profile.ShouldUseConstructor , Profile.SourceMemberNamingConvention , Profile.DestinationMemberNamingConvention , Profile.ValueTransformers , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

AuthorHaikuMapping()

Initializes a new instance of the AuthorHaikuMapping class.

```
public AuthorHaikuMapping()
```

Remarks

This constructor creates mappings between AuthorHaiku and AuthorHaikuDto.

Class AuthorMapping

Namespace: Haiku.API.Mapping

Assembly: Haiku.API.dll

```
public class AuthorMapping : Profile, IProfileExpression, IProfileConfiguration
```

Inheritance

object ← Profile ← AuthorMapping

Implements

IProfileExpression, IProfileConfiguration

Inherited Members

Profile.DisableConstructorMapping() , Profile.CreateProjection<TSource, TDestination>() , Profile.CreateProjection<TSource, TDestination>(MemberList) , Profile.CreateMap<TSource, TDestination>() , Profile.CreateMap<TSource, TDestination>(MemberList) , Profile.CreateMap(Type, Type) , Profile.CreateMap(Type, Type, MemberList) , Profile.ClearPrefixes() , Profile.ReplaceMemberName(string, string) , Profile.RecognizePrefixes(params string[]) , Profile.RecognizePostfixes(params string[]) , Profile.RecognizeDestinationPrefixes(params string[]) , Profile.RecognizeDestinationPostfixes(params string[]) , Profile.AddGlobalIgnore(string) , Profile.IncludeSourceExtensionMethods(Type) , Profile.ProfileName , Profile.AllowNullDestinationValues , Profile.AllowNullCollections , Profile.ShouldMapProperty , Profile.ShouldMapField , Profile.ShouldMapMethod , Profile.ShouldUseConstructor , Profile.SourceMemberNamingConvention , Profile.DestinationMemberNamingConvention , Profile.ValueTransformers , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

AuthorMapping()

Initializes a new instance of the AuthorMapping class.

```
public AuthorMapping()
```

Remarks

This constructor creates mappings between Author and AuthorDto.

Class UserHaikuMapping

Namespace: Haiku.API.Mapping

Assembly: Haiku.API.dll

```
public class UserHaikuMapping : Profile, IProfileExpression, IProfileConfiguration
```

Inheritance

object ← Profile ← UserHaikuMapping

Implements

IProfileExpression, IProfileConfiguration

Inherited Members

Profile.DisableConstructorMapping() , Profile.CreateProjection<TSource, TDestination>() , Profile.CreateProjection<TSource, TDestination>(MemberList) , Profile.CreateMap<TSource, TDestination>() , Profile.CreateMap<TSource, TDestination>(MemberList) , Profile.CreateMap(Type, Type) , Profile.CreateMap(Type, Type, MemberList) , Profile.ClearPrefixes() , Profile.ReplaceMemberName(string, string) , Profile.RecognizePrefixes(params string[]) , Profile.RecognizePostfixes(params string[]) , Profile.RecognizeDestinationPrefixes(params string[]) , Profile.RecognizeDestinationPostfixes(params string[]) , Profile.AddGlobalIgnore(string) , Profile.IncludeSourceExtensionMethods(Type) , Profile.ProfileName , Profile.AllowNullDestinationValues , Profile.AllowNullCollections , Profile.ShouldMapProperty , Profile.ShouldMapField , Profile.ShouldMapMethod , Profile.ShouldUseConstructor , Profile.SourceMemberNamingConvention , Profile.DestinationMemberNamingConvention , Profile.ValueTransformers , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

UserHaikuMapping()

Initializes a new instance of the UserHaikuMapping class.

```
public UserHaikuMapping()
```

Remarks

This constructor creates mappings between UserHaiku and UserHaikuDto.

Class UserMapping

Namespace: Haiku.API.Mapping

Assembly: Haiku.API.dll

```
public class UserMapping : Profile, IProfileExpression, IProfileConfiguration
```

Inheritance

object ← Profile ← UserMapping

Implements

IProfileExpression, IProfileConfiguration

Inherited Members

Profile.DisableConstructorMapping() , Profile.CreateProjection<TSource, TDestination>() , Profile.CreateProjection<TSource, TDestination>(MemberList) , Profile.CreateMap<TSource, TDestination>() , Profile.CreateMap<TSource, TDestination>(MemberList) , Profile.CreateMap(Type, Type) , Profile.CreateMap(Type, Type, MemberList) , Profile.ClearPrefixes() , Profile.ReplaceMemberName(string, string) , Profile.RecognizePrefixes(params string[]) , Profile.RecognizePostfixes(params string[]) , Profile.RecognizeDestinationPrefixes(params string[]) , Profile.RecognizeDestinationPostfixes(params string[]) , Profile.AddGlobalIgnore(string) , Profile.IncludeSourceExtensionMethods(Type) , Profile.ProfileName , Profile.AllowNullDestinationValues , Profile.AllowNullCollections , Profile.ShouldMapProperty , Profile.ShouldMapField , Profile.ShouldMapMethod , Profile.ShouldUseConstructor , Profile.SourceMemberNamingConvention , Profile.DestinationMemberNamingConvention , Profile.ValueTransformers , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

UserMapping()

Initializes a new instance of the UserMapping class.

```
public UserMapping()
```

Remarks

This constructor creates mappings between:

- User and UserDto
- RegisterDto and User
- LoginDto and User

Class UserProfileMapping

Namespace: Haiku.API.Mapping

Assembly: Haiku.API.dll

```
public class UserProfileMapping : Profile, IProfileExpression, IProfileConfiguration
```

Inheritance

object ← Profile ← UserProfileMapping

Implements

IProfileExpression, IProfileConfiguration

Inherited Members

Profile.DisableConstructorMapping() , Profile.CreateProjection<TSource, TDestination>() , Profile.CreateProjection<TSource, TDestination>(MemberList) , Profile.CreateMap<TSource, TDestination>() , Profile.CreateMap<TSource, TDestination>(MemberList) , Profile.CreateMap(Type, Type) , Profile.CreateMap(Type, Type, MemberList) , Profile.ClearPrefixes() , Profile.ReplaceMemberName(string, string) , Profile.RecognizePrefixes(params string[]) , Profile.RecognizePostfixes(params string[]) , Profile.RecognizeDestinationPrefixes(params string[]) , Profile.RecognizeDestinationPostfixes(params string[]) , Profile.AddGlobalIgnore(string) , Profile.IncludeSourceExtensionMethods(Type) , Profile.ProfileName , Profile.AllowNullDestinationValues , Profile.AllowNullCollections , Profile.ShouldMapProperty , Profile.ShouldMapField , Profile.ShouldMapMethod , Profile.ShouldUseConstructor , Profile.SourceMemberNamingConvention , Profile.DestinationMemberNamingConvention , Profile.ValueTransformers , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

UserProfileMapping()

Initializes a new instance of the UserProfileMapping class.

```
public UserProfileMapping()
```

Remarks

This constructor creates mappings between UserProfile and UserProfileDto.

Namespace Haiku.API.Migrations

Classes

Test

A base class inherited by each EF Core migration.

Class Test

Namespace: Haiku.API.Migrations

Assembly: Haiku.API.dll

A base class inherited by each EF Core migration.

```
[DbContext(typeof(HaikuAPIContext))]
[Migration("20241020163518_Test")]
public class Test : Migration
```

Inheritance

object ← Migration ← Test

Inherited Members

Migration.InitialDatabase , Migration.TargetModel , Migration.UpOperations , Migration.DownOperations , Migration.ActiveProvider , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Remarks

See Database migrations for more information and examples.

Methods

BuildTargetModel(ModelBuilder)

Implemented to build the TargetModel.

```
protected override void BuildTargetModel(ModelBuilder modelBuilder)
```

Parameters

`modelBuilder` ModelBuilder

The ModelBuilder to use to build the model.

Remarks

See Database migrations for more information and examples.

Down(MigrationBuilder)

Builds the operations that will migrate the database 'down'.

```
protected override void Down(MigrationBuilder migrationBuilder)
```

Parameters

`migrationBuilder` MigrationBuilder

The MigrationBuilder that will build the operations.

Remarks

That is, builds the operations that will take the database from the state left in by this migration so that it returns to the state that it was in before this migration was applied.

This method must be overridden in each class that inherits from Migration if both 'up' and 'down' migrations are to be supported. If it is not overridden, then calling it will throw and it will not be possible to migrate in the 'down' direction.

See Database migrations for more information and examples.

Up(MigrationBuilder)

Builds the operations that will migrate the database 'up'.

```
protected override void Up(MigrationBuilder migrationBuilder)
```

Parameters

`migrationBuilder` MigrationBuilder

The MigrationBuilder that will build the operations.

Remarks

That is, builds the operations that will take the database from the state left in by the previous migration so that it is up-to-date with regard to this migration.

This method must be overridden in each class that inherits from Migration.

See Database migrations for more information and examples.

Namespace Haiku.API.Models

Classes

Author

AuthorHaiku

ErrorDetails

JWToken

JwtSettings

Login

PaginationMetaData

Register

Role

User

UserHaiku

UserProfile

Class Author

Namespace: Haiku.API.Models

Assembly: Haiku.API.dll

```
public class Author
```

Inheritance

object ← Author

Inherited Members

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

Properties

AuthorHaikus

```
public virtual ICollection<AuthorHaiku> AuthorHaikus { get; set; }
```

Property Value

ICollection<AuthorHaiku>

Bio

```
[Column]
[StringLength(1000, ErrorMessage = "Bio length can't be more than 300 characters.")]
[MinLength(4, ErrorMessage = "Name length must be at least 4 characters.")]
public string? Bio { get; set; }
```

Property Value

string

Id

```
[Key]
[Column]
public long Id { get; set; }
```

Property Value

long

Name

```
[Required]
[Column]
[StringLength(50, ErrorMessage = "Name length can't be more than 50 characters.")]
[MinLength(2, ErrorMessage = "Name length must be at least 2 characters.")]
public required string Name { get; set; }
```

Property Value

string

Class AuthorHaiku

Namespace: Haiku.API.Models

Assembly: Haiku.API.dll

```
public class AuthorHaiku
```

Inheritance

object ← AuthorHaiku

Inherited Members

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

Properties

Author

```
[ForeignKey("AuthorId")]
public Author? Author { get; set; }
```

Property Value

Author

AuthorId

```
public long? AuthorId { get; set; }
```

Property Value

long?

Id

```
[Key]
[Column]
public long Id { get; set; }
```

Property Value

long

LineOne

```
[Required]
[Column]
[SyllableCount(5, ErrorMessage = "Must be five syllables")]
public required string LineOne { get; set; }
```

Property Value

string

LineThree

```
[Required]
[Column]
[SyllableCount(5, ErrorMessage = "Must be five syllables")]
public required string LineThree { get; set; }
```

Property Value

string

LineTwo

```
[Required]
[Column]
[SyllableCount(7, ErrorMessage = "Must be seven syllables")]
public required string LineTwo { get; set; }
```

Property Value

string

Title

```
[Column]
[StringLength(100, ErrorMessage = "Title length can't be more than 100.")]
[MinLength(4, ErrorMessage = "Title length must be at least 4 characters.")]
public string? Title { get; set; }
```

Property Value

string

Class ErrorDetails

Namespace: Haiku.API.Models

Assembly: Haiku.API.dll

```
public class ErrorDetails
```

Inheritance

object ← ErrorDetails

Inherited Members

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

Properties

Message

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

Property Value

string

StatusCode

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

Property Value

int

Methods

ToXml()

```
public string ToXml()
```

Returns

string

Class JWToken

Namespace: Haiku.API.Models

Assembly: Haiku.API.dll

```
public class JWToken
```

Inheritance

object ← JWToken

Inherited Members

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

Properties

ExpiresIn

```
[Required]
public required string ExpiresIn { get; set; }
```

Property Value

string

Id

```
public long Id { get; set; }
```

Property Value

long

Token

```
[Required]
public required string Token { get; set; }
```

Property Value

string

UserId

```
[Required]
public long UserId { get; set; }
```

Property Value

long

Class JwtSettings

Namespace: Haiku.API.Models

Assembly: Haiku.API.dll

```
public class JwtSettings
```

Inheritance

object ← JwtSettings

Inherited Members

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

Properties

Audience

```
public required string Audience { get; set; }
```

Property Value

string

Issuer

```
public required string Issuer { get; set; }
```

Property Value

string

Key

```
public required string Key { get; set; }
```

Property Value

string

Class Login

Namespace: Haiku.API.Models

Assembly: Haiku.API.dll

```
public class Login
```

Inheritance

object ← Login

Inherited Members

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

Properties

Id

```
[Key]
[Column]
public long Id { get; set; }
```

Property Value

long

Password

```
[Required]
[Column]
[StringLength(30, ErrorMessage = "Password length can't be more than 30 characters.")]
[MinLength(8, ErrorMessage = "Password length must be at least 8 characters.")]
public required string Password { get; set; }
```

Property Value

string

Username

```
[Required]
[Column]
[StringLength(20, ErrorMessage = "Username length can't be more than 20 characters.")]
[MinLength(4, ErrorMessage = "Username length must be at least 4 characters.")]
public required string Username { get; set; }
```

Property Value

string

Class PaginationMetaData

Namespace: Haiku.API.Models

Assembly: Haiku.API.dll

```
public class PaginationMetaData
```

Inheritance

object ← PaginationMetaData

Inherited Members

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

Properties

CurrentPage

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

Property Value

int

PageSize

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

Property Value

int

TotalCount

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

Property Value

int

TotalPages

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

Property Value

int

Class Register

Namespace: Haiku.API.Models

Assembly: Haiku.API.dll

```
public class Register
```

Inheritance

object ← Register

Inherited Members

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

Properties

ConfirmPassword

```
[Required]
[Compare("Password", ErrorMessage = "Passwords do not match.")]
public required string ConfirmPassword { get; set; }
```

Property Value

string

Id

```
public long Id { get; set; }
```

Property Value

long

Password

```
[Required]
[StringLength(30, ErrorMessage = "Password length can't be more than 30 characters.")]
[MinLength(8, ErrorMessage = "Password length must be at least 8 characters.")]
public required string Password { get; set; }
```

Property Value

string

Username

```
[Required]
[StringLength(20, ErrorMessage = "Username length can't be more than 20 characters.")]
[MinLength(4, ErrorMessage = "Username length must be at least 4 characters.")]
public required string Username { get; set; }
```

Property Value

string

Class Role

Namespace: Haiku.API.Models

Assembly: Haiku.API.dll

```
public class Role
```

Inheritance

object ← Role

Inherited Members

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

Properties

Id

```
[Key]
[Column]
public long Id { get; set; }
```

Property Value

long

Title

```
[Required]
[Column]
public required string Title { get; set; }
```

Property Value

string

Users

```
public virtual ICollection<User> Users { get; set; }
```

Property Value

ICollection <User>

Class User

Namespace: Haiku.API.Models

Assembly: Haiku.API.dll

```
public class User
```

Inheritance

object ← User

Inherited Members

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

Properties

Id

```
[Key]
[Column]
public long Id { get; set; }
```

Property Value

long

Password

```
[Required]
[Column(TypeName = "nvarchar(256)")]
[StringLength(20, ErrorMessage = "Password length can't be more than 20 characters.")]
[MinLength(8, ErrorMessage = "Password length must be at least 8 characters.")]
public required string Password { get; set; }
```

Property Value

string

Profile

```
public virtual UserProfile? Profile { get; set; }
```

Property Value

UserProfile

RoleId

```
[Required]
public long RoleId { get; set; }
```

Property Value

long

UserHaikus

```
public virtual ICollection<UserHaiku> UserHaikus { get; set; }
```

Property Value

ICollection<UserHaiku>

UserRole

```
[ForeignKey("RoleId")]
public virtual Role? UserRole { get; set; }
```

Property Value

Role

Username

```
[Required]
[Column]
[StringLength(20, ErrorMessage = "Username length can't be more than 20 characters.")]
[MinLength(4, ErrorMessage = "Username length must be at least 4 characters.")]
public required string Username { get; set; }
```

Property Value

string

Class UserHaiku

Namespace: Haiku.API.Models

Assembly: Haiku.API.dll

```
public class UserHaiku
```

Inheritance

object ← UserHaiku

Inherited Members

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

Properties

Id

```
[Key]
[Column]
public long Id { get; set; }
```

Property Value

long

LineOne

```
[Required]
[Column]
[SyllableCount(5, ErrorMessage = "Must be five syllables")]
public required string LineOne { get; set; }
```

Property Value

string

LineThree

```
[Required]
[Column]
[SyllableCount(5, ErrorMessage = "Must be five syllables")]
public required string LineThree { get; set; }
```

Property Value

string

LineTwo

```
[Required]
[Column]
[SyllableCount(7, ErrorMessage = "Must be seven syllables")]
public required string LineTwo { get; set; }
```

Property Value

string

Title

```
[Column]
[StringLength(100, ErrorMessage = "Title length can't be more than 100.")]
[MinLength(4, ErrorMessage = "Title length must be at least 4 characters.")]
public string? Title { get; set; }
```

Property Value

string

User

```
[ForeignKey("UserId")]
public User? User { get; set; }
```

Property Value

User

UserId

```
[Required]
public long UserId { get; set; }
```

Property Value

long

Class UserProfile

Namespace: Haiku.API.Models

Assembly: Haiku.API.dll

```
public class UserProfile
```

Inheritance

object ← UserProfile

Inherited Members

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

Properties

Bio

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

Property Value

string

Id

```
[Key]
[Column]
public long Id { get; set; }
```

Property Value

long

ImageUrl

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

Property Value

string

User

```
[ForeignKey("UserId")]
public virtual User? User { get; set; }
```

Property Value

User

UserId

```
[Required]
public long UserId { get; set; }
```

Property Value

long

Namespace Haiku.API.Repositories.Author HaikuRepositories

Classes

AuthorHaikuRepository

Interfaces

IAuthorHaikuRepository

Class AuthorHaikuRepository

Namespace: Haiku.API.Repositories.AuthorHaikuRepositories

Assembly: Haiku.API.dll

```
public class AuthorHaikuRepository : IAuthorHaikuRepository
```

Inheritance

object ← AuthorHaikuRepository

Implements

IAuthorHaikuRepository

Inherited Members

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

Constructors

AuthorHaikuRepository(HaikuAPIContext)

```
public AuthorHaikuRepository(HaikuAPIContext context)
```

Parameters

`context` HaikuAPIContext

Methods

AddAuthorHaikuAsync(AuthorHaiku)

Adds a new AuthorHaiku to the database asynchronously.

```
public Task<AuthorHaiku> AddAuthorHaikuAsync(AuthorHaiku newHaiku)
```

Parameters

`newHaiku` AuthorHaiku

The AuthorHaiku entity to be added.

Returns

Task<AuthorHaiku>

A task that represents the asynchronous operation. The task result contains the added AuthorHaiku entity.

AuthorHaikuExistsByIdAsync(long)

Checks if an AuthorHaiku entity exists by its ID asynchronously.

```
public Task<bool> AuthorHaikuExistsByIdAsync(long authorHaikuId)
```

Parameters

`authorHaikuId` long

The ID of the AuthorHaiku entity to check for existence.

Returns

Task<bool>

A task that represents the asynchronous operation. The task result contains a boolean value indicating whether the AuthorHaiku exists.

DeleteAuthorHaikuByIdAsync(long)

Deletes an AuthorHaiku entity by its ID asynchronously.

```
public Task<int> DeleteAuthorHaikuByIdAsync(long authorHaikuId)
```

Parameters

`authorHaikuId` long

The ID of the AuthorHaiku entity to be deleted.

Returns

Task<int>

A task that represents the asynchronous operation. The task result contains the number of affected rows in the database.

GetAuthorHaikuByIdAsync(long)

Retrieves an AuthorHaiku entity by its unique identifier.

```
public Task<AuthorHaiku?> GetAuthorHaikuByIdAsync(long authorHaikuId)
```

Parameters

`authorHaikuId` long

The unique identifier of the AuthorHaiku to retrieve.

Returns

Task<AuthorHaiku>

A task that represents the asynchronous operation. The task result contains the AuthorHaiku entity if found; otherwise, `null`.

GetPaginatedAuthorHaikusAsync(int, int, string)

Retrieves a paginated list of AuthorHaiku entities.

```
public Task<IEnumerable<AuthorHaiku>> GetPaginatedAuthorHaikusAsync(int pageNumber, int
pageSize, string searchOption)
```

Parameters

`pageNumber` int

The number of the page to retrieve. Must be greater than zero.

`pageSize` int

The number of items per page. Must be greater than zero.

`searchOption` string

An optional search term to filter the haikus by title.

Returns

Task<IEnumerable<AuthorHaiku>>

A task that represents the asynchronous operation. The task result contains a collection of Author Haiku entities.

GetPaginatedAuthorHaikusByAuthorIdAsync(long, int, int, string)

Retrieves a paginated list of AuthorHaiku entities for a specific author.

```
public Task<IEnumerable<AuthorHaiku>> GetPaginatedAuthorHaikusByAuthorIdAsync(long authorId,
int pageNumber, int pageSize, string searchOption)
```

Parameters

`authorId` long

The ID of the author whose haikus are to be retrieved. Must be a valid identifier.

`pageNumber` int

The number of the page to retrieve. Must be greater than zero.

`pageSize` int

The number of items per page. Must be greater than zero.

`searchOption` string

An optional search term to filter the haikus by title.

Returns

Task<IEnumerable<AuthorHaiku>>

A task that represents the asynchronous operation. The task result contains a collection of AuthorHaiku entities.

GetTotalAuthorHaikusAsync(string)

Retrieves the total count of AuthorHaiku entities in the database, optionally filtered by a search term for the title.

```
public Task<int> GetTotalAuthorHaikusAsync(string searchOption)
```

Parameters

`searchOption` string

An optional search term to filter the haikus by title.

Returns

Task<int>

A task that represents the asynchronous operation. The task result contains the total count of AuthorHaiku entities.

GetTotalAuthorHaikusByAuthorIdAsync(long, string)

Retrieves the total count of AuthorHaiku entities associated with a specific author, optionally filtered by a search term for the title.

```
public Task<int> GetTotalAuthorHaikusByAuthorIdAsync(long authorId, string searchOption)
```

Parameters

`authorId` long

The ID of the author whose haikus are being counted. Must be a valid identifier.

`searchOption` string

An optional search term to filter the haikus by title.

Returns

Task<int>

A task that represents the asynchronous operation. The task result contains the total count of Author Haiku entities.

UpdateAuthorHaikuAsync(AuthorHaiku)

Updates an existing AuthorHaiku in the database asynchronously.

```
public Task<int> UpdateAuthorHaikuAsync(AuthorHaiku updatedAuthorHaiku)
```

Parameters

`updatedAuthorHaiku` AuthorHaiku

The AuthorHaiku entity with updated values.

Returns

Task<int>

A task that represents the asynchronous operation. The task result contains the number of state entries written to the database.

UpdateAuthorHaikusToUnknownAuthorAsync(long, long)

Updates the author ID of AuthorHaiku entities that have a specific old author ID or are currently associated with an unknown author (null) to a new author ID asynchronously.

```
public Task<int> UpdateAuthorHaikusToUnknownAuthorAsync(long oldAuthorId, long newAuthorId)
```

Parameters

`oldAuthorId` long

The ID of the author whose haikus are being updated.

`newAuthorId` long

The ID of the new author to associate with the haikus.

Returns

Task<int>

A task that represents the asynchronous operation. The task result contains the number of affected rows in the database.

Interface IAuthorHaikuRepository

Namespace: Haiku.API.Repositories.AuthorHaikuRepositories

Assembly: Haiku.API.dll

```
public interface IAuthorHaikuRepository
```

Methods

AddAuthorHaikuAsync(AuthorHaiku)

```
Task<AuthorHaiku> AddAuthorHaikuAsync(AuthorHaiku AuthorHaiku)
```

Parameters

`AuthorHaiku` AuthorHaiku

Returns

Task<AuthorHaiku>

AuthorHaikuExistsByIdAsync(long)

```
Task<bool> AuthorHaikuExistsByIdAsync(long AuthorHaikuId)
```

Parameters

`AuthorHaikuId` long

Returns

Task<bool>

DeleteAuthorHaikuByIdAsync(long)

```
Task<int> DeleteAuthorHaikuByIdAsync(long AuthorHaikuId)
```

Parameters

`AuthorHaikuId` long

Returns

Task<int>

GetAuthorHaikuByIdAsync(long)

```
Task<AuthorHaiku?> GetAuthorHaikuByIdAsync(long AuthorHaikuId)
```

Parameters

`AuthorHaikuId` long

Returns

Task<AuthorHaiku>

GetPaginatedAuthorHaikusAsync(int, int, string)

```
Task<IEnumerable<AuthorHaiku>> GetPaginatedAuthorHaikusAsync(int pageNumber, int pageSize,
string searchOption)
```

Parameters

`pageNumber` int

`pageSize` int

`searchOption` string

Returns

Task<IEnumerable<AuthorHaiku>>

GetPaginatedAuthorHaikusByAuthorIdAsync(long, int, int, string)

```
Task<IEnumerable<AuthorHaiku>> GetPaginatedAuthorHaikusByAuthorIdAsync(long authorId, int
pageNumber, int pageSize, string searchOption)
```

Parameters

authorId long

pageNumber int

pageSize int

searchOption string

Returns

Task<IEnumerable<AuthorHaiku>>

GetTotalAuthorHaikusAsync(string)

```
Task<int> GetTotalAuthorHaikusAsync(string searchOption)
```

Parameters

searchOption string

Returns

Task<int>

GetTotalAuthorHaikusByAuthorIdAsync(long, string)

```
Task<int> GetTotalAuthorHaikusByAuthorIdAsync(long authorId, string searchOption)
```

Parameters

`authorId` long

`searchOption` string

Returns

Task<int>

UpdateAuthorHaikuAsync(AuthorHaiku)

```
Task<int> UpdateAuthorHaikuAsync(AuthorHaiku AuthorHaiku)
```

Parameters

`AuthorHaiku` AuthorHaiku

Returns

Task<int>

UpdateAuthorHaikusToUnknownAuthorAsync(long, long)

```
Task<int> UpdateAuthorHaikusToUnknownAuthorAsync(long oldAuthorId, long newAuthorId)
```

Parameters

`oldAuthorId` long

`newAuthorId` long

Returns

Task<int>

Namespace Haiku.API.Repositories.AuthorRepositories

Classes

AuthorRepository

Interfaces

IAuthorRepository

Class AuthorRepository

Namespace: Haiku.API.Repositories.AuthorRepositories

Assembly: Haiku.API.dll

```
public class AuthorRepository : IAuthorRepository
```

Inheritance

object ← AuthorRepository

Implements

IAuthorRepository

Inherited Members

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

Constructors

AuthorRepository(HaikuAPIContext)

```
public AuthorRepository(HaikuAPIContext context)
```

Parameters

`context` HaikuAPIContext

Methods

AddAuthorAsync(Author)

Adds a new Author to the database asynchronously.

```
public Task<Author> AddAuthorAsync(Author newAuthor)
```

Parameters

`newAuthor` Author

The Author entity to be added.

Returns

Task<Author>

A task that represents the asynchronous operation. The task result contains the added Author entity.

AuthorExistsByIdAsync(long)

Checks if an Author entity exists by its ID asynchronously.

```
public Task<bool> AuthorExistsByIdAsync(long authorId)
```

Parameters

`authorId` long

The ID of the Author entity to check for existence.

Returns

Task<bool>

A task that represents the asynchronous operation. The task result contains a boolean value indicating whether the Author exists.

DeleteAuthorByIdAsync(long)

Deletes an existing Author entry.

```
public Task<int> DeleteAuthorByIdAsync(long authorId)
```

Parameters

`authorId` long

The ID of the Author entry to delete.

Returns

Task<int>

Integer result indicating number of rows affected, if Author was deleted.

GetAuthorByIdAsync(long)

Retrieves an Author entity by its unique identifier.

```
public Task<Author?> GetAuthorByIdAsync(long authorId)
```

Parameters

`authorId` long

Returns

Task<Author>

A task that represents the asynchronous operation. The task result contains the Author entity if found; otherwise, `null`.

GetPaginatedAuthorsAsync(int, int, string)

Retrieves a paginated list of Author entities.

```
public Task<IEnumerable<Author>> GetPaginatedAuthorsAsync(int pageNumber, int pageSize,
string searchOption)
```

Parameters

`pageNumber` int

The page number. Must be greater than 0.

`pageSize` int

The number of Authors per page. Must be greater than 0.

`searchOption` string

An optional search term to filter the Author entities by name.

Returns

Task<IEnumerable<Author>>

IEnumerable<T> containing Authors, with pageSize amount per page. Returns an empty collection if no Authors are found.

GetTotalAuthorsAsync(string)

Retrieves the total count of Author entities in the database, optionally filtered by a search term for the name.

```
public Task<int> GetTotalAuthorsAsync(string searchOption)
```

Parameters

`searchOption` string

An optional search term to filter the Author entities by name.

Returns

Task<int>

A task that represents the asynchronous operation. The task result contains the total count of Author entities.

UpdateAuthorAsync(Author)

Updates an existing Author entry.

```
public Task<int> UpdateAuthorAsync(Author updatedAuthor)
```

Parameters

`updatedAuthor` Author

The Author entry to update. Must not be null.

Returns

Task<int>

Integer result indicating number of rows affected, if Author was updated.

Interface IAuthorRepository

Namespace: Haiku.API.Repositories.AuthorRepositories

Assembly: Haiku.API.dll

```
public interface IAuthorRepository
```

Methods

AddAuthorAsync(Author)

```
Task<Author> AddAuthorAsync(Author newAuthor)
```

Parameters

`newAuthor` Author

Returns

Task<Author>

AuthorExistsByIdAsync(long)

```
Task<bool> AuthorExistsByIdAsync(long authorId)
```

Parameters

`authorId` long

Returns

Task<bool>

DeleteAuthorByIdAsync(long)

```
Task<int> DeleteAuthorByIdAsync(long authorId)
```

Parameters

`authorId` long

Returns

Task<int>

GetAuthorByIdAsync(long)

```
Task<Author?> GetAuthorByIdAsync(long authorId)
```

Parameters

`authorId` long

Returns

Task<Author>

GetPaginatedAuthorsAsync(int, int, string)

```
Task<IEnumerable<Author>> GetPaginatedAuthorsAsync(int pageNumber, int pageSize,
string searchOption)
```

Parameters

`pageNumber` int

`pageSize` int

`searchOption` string

Returns

Task<IEnumerable<Author>>

GetTotalAuthorsAsync(string)

```
Task<int> GetTotalAuthorsAsync(string searchOption)
```

Parameters

`searchOption` string

Returns

Task<int>

UpdateAuthorAsync(Author)

```
Task<int> UpdateAuthorAsync(Author updatedAuthor)
```

Parameters

`updatedAuthor` Author

Returns

Task<int>

Namespace Haiku.API.Repositories.RoleRepositories

Classes

RoleRepository

Interfaces

IRoleRepository

Interface IRoleRepository

Namespace: Haiku.API.Repositories.RoleRepositories

Assembly: Haiku.API.dll

```
public interface IRoleRepository
```

Methods

GetUserRoleAsync(long)

```
Task<Role?> GetUserRoleAsync(long userId)
```

Parameters

`userId` long

Returns

Task<Role>

Class RoleRepository

Namespace: Haiku.API.Repositories.RoleRepositories

Assembly: Haiku.API.dll

```
public class RoleRepository : IRoleRepository
```

Inheritance

object ← RoleRepository

Implements

IRoleRepository

Inherited Members

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

Constructors

RoleRepository(HaikuAPIContext)

```
public RoleRepository(HaikuAPIContext context)
```

Parameters

`context` HaikuAPIContext

Methods

GetUserRoleAsync(long)

Retrieves the Role of a User asynchronously based on the provided User ID.

```
public Task<Role?> GetUserRoleAsync(long userId)
```

Parameters

`userId` long

The ID of the User whose Role is to be retrieved.

Returns

Task<Role>

A task that represents the asynchronous operation. The task result contains the Role of the User, or `null` if the User or their Role does not exist.

Namespace Haiku.API.Repositories.UserHaiku Repositories

Classes

UserHaikuRepository

Interfaces

IUserHaikuRepository

Interface IUserHaikuRepository

Namespace: Haiku.API.Repositories.UserHaikuRepositories

Assembly: Haiku.API.dll

```
public interface IUserHaikuRepository
```

Methods

AddUserHaikuAsync(UserHaiku)

```
Task<UserHaiku> AddUserHaikuAsync(UserHaiku UserHaiku)
```

Parameters

`UserHaiku` UserHaiku

Returns

Task<UserHaiku>

DeleteUserHaikuByIdAsync(long)

```
Task<int> DeleteUserHaikuByIdAsync(long UserHaikuId)
```

Parameters

`UserHaikuId` long

Returns

Task<int>

GetPaginatedUserHaikusAsync(int, int, string)

```
Task<IEnumerable<UserHaiku>> GetPaginatedUserHaikusAsync(int pageNumber, int pageSize,
string searchOption)
```

Parameters

`pageNumber` int

`pageSize` int

`searchOption` string

Returns

Task<IEnumerable<UserHaiku>>

GetPaginatedUserHaikusByUserIdAsync(long, int, int, string)

```
Task<IEnumerable<UserHaiku>> GetPaginatedUserHaikusByUserIdAsync(long userId, int
pageNumber, int pageSize, string searchOption)
```

Parameters

`userId` long

`pageNumber` int

`pageSize` int

`searchOption` string

Returns

Task<IEnumerable<UserHaiku>>

GetTotalUserHaikusAsync(string)

```
Task<int> GetTotalUserHaikusAsync(string searchOption)
```

Parameters

`searchOption` string

Returns

Task<int>

GetTotalUserHaikusByUserIdAsync(long, string)

```
Task<int> GetTotalUserHaikusByUserIdAsync(long userId, string searchOption)
```

Parameters

`userId` long

`searchOption` string

Returns

Task<int>

GetUserHaikuByIdAsync(long)

```
Task<UserHaiku?> GetUserHaikuByIdAsync(long UserHaikuId)
```

Parameters

`UserHaikuId` long

Returns

Task<UserHaiku>

GetUserHaikusByUserIdForDeleteAsync(long)

```
Task<IEnumerable<UserHaiku>> GetUserHaikusByUserIdForDeleteAsync(long userId)
```

Parameters

`userId` long

Returns

Task<IEnumerable<UserHaiku>>

UpdateUserHaikuAsync(UserHaiku)

```
Task<int> UpdateUserHaikuAsync(UserHaiku UserHaiku)
```

Parameters

`UserHaiku` UserHaiku

Returns

Task<int>

UserHaikuExistsByIdAsync(long)

```
Task<bool> UserHaikuExistsByIdAsync(long UserHaikuId)
```

Parameters

`UserHaikuId` long

Returns

Task<bool>

Class UserHaikuRepository

Namespace: Haiku.API.Repositories.UserHaikuRepositories

Assembly: Haiku.API.dll

```
public class UserHaikuRepository : IUserHaikuRepository
```

Inheritance

object ← UserHaikuRepository

Implements

IUserHaikuRepository

Inherited Members

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

Constructors

UserHaikuRepository(HaikuAPIContext)

```
public UserHaikuRepository(HaikuAPIContext context)
```

Parameters

`context` HaikuAPIContext

Methods

AddUserHaikuAsync(UserHaiku)

Adds a new UserHaiku to the database asynchronously.

```
public Task<UserHaiku> AddUserHaikuAsync(UserHaiku haiku)
```

Parameters

`haiku` UserHaiku

Returns

Task<UserHaiku>

A task that represents the asynchronous operation. The task result contains the added UserHaiku entity.

DeleteUserHaikuByIdAsync(long)

Deletes an UserHaiku entity by its ID asynchronously.

```
public Task<int> DeleteUserHaikuByIdAsync(long userHaikuId)
```

Parameters

`userHaikuId` long

The ID of the UserHaiku entity to be deleted.

Returns

Task<int>

A task that represents the asynchronous operation. The task result contains the number of affected rows in the database.

GetPaginatedUserHaikusAsync(int, int, string)

Retrieves a paginated list of UserHaiku entities.

```
public Task<IEnumerable<UserHaiku>> GetPaginatedUserHaikusAsync(int pageNumber, int
pageSize, string searchOption)
```

Parameters

`pageNumber` int

The number of the page to retrieve. Must be greater than zero.

`pageSize` int

The number of items per page. Must be greater than zero.

`searchOption` string

An optional search term to filter the UserHaiku by title.

Returns

Task<IEnumerable<UserHaiku>>

A task that represents the asynchronous operation. The task result contains a collection of UserHaiku entities.

GetPaginatedUserHaikusByUserIdAsync(long, int, int, string)

Retrieves a paginated list of UserHaiku entities by user ID.

```
public Task<IEnumerable<UserHaiku>> GetPaginatedUserHaikusByUserIdAsync(long userId, int
pageNumber, int pageSize, string searchOption)
```

Parameters

`userId` long

The user identification to filter the UserHaiku.

`pageNumber` int

The number of the page to retrieve. Must be greater than zero.

`pageSize` int

The number of items per page. Must be greater than zero.

`searchOption` string

An optional search term to filter the UserHaiku by title.

Returns

Task<IEnumerable<UserHaiku>>

A task that represents the asynchronous operation. The task result contains a collection of UserHaiku entities.

GetTotalUserHaikusAsync(string)

Retrieves the total count of UserHaiku entities in the database, optionally filtered by a search term for the title.

```
public Task<int> GetTotalUserHaikusAsync(string searchOption)
```

Parameters

`searchOption` string

An optional search term to filter the UserHaiku by title.

Returns

Task<int>

A task that represents the asynchronous operation. The task result contains the total count of User Haiku entities.

GetTotalUserHaikusByUserIdAsync(long, string)

Retrieves the total count of UserHaiku entities in the database by the User identifier, optionally filtered by a search term for the title.

```
public Task<int> GetTotalUserHaikusByUserIdAsync(long userId, string searchOption)
```

Parameters

`userId` long

The user identification to filter the UserHaiku.

`searchOption` string

An optional search term to filter the UserHaiku by title.

Returns

Task<int>

A task that represents the asynchronous operation. The task result contains the total count of UserHaiku entities.

GetUserHaikuByIdAsync(long)

Retrieves an UserHaiku entity by its unique identifier.

```
public Task<UserHaiku?> GetUserHaikuByIdAsync(long haikuId)
```

Parameters

`haikuId` long

Returns

Task<UserHaiku>

A task that represents the asynchronous operation. The task result contains the UserHaiku entity if found; otherwise, `null`.

GetUserHaikusByUserIdForDeleteAsync(long)

Retrieves a collection of UserHaiku entities associated with a specified user ID for deletion purposes asynchronously.

```
public Task<IEnumerable<UserHaiku>> GetUserHaikusByUserIdForDeleteAsync(long userId)
```

Parameters

`userId` long

The ID of the User whose haikus are to be retrieved.

Returns

Task<IEnumerable<UserHaiku>>

A task that represents the asynchronous operation. The task result contains an IEnumerable<T> collection of the User's UserHaiku.

UpdateUserHaikuAsync(UserHaiku)

Updates an existing UserHaiku in the database asynchronously.

```
public Task<int> UpdateUserHaikuAsync(UserHaiku updatedUserHaiku)
```

Parameters

`updatedUserHaiku` UserHaiku

The UserHaiku entity with updated values.

Returns

Task<int>

A task that represents the asynchronous operation. The task result contains the number of state entries written to the database.

UserHaikuExistsByIdAsync(long)

Checks if an UserHaiku entity exists by its ID asynchronously.

```
public Task<bool> UserHaikuExistsByIdAsync(long userHaikuId)
```

Parameters

`userHaikuId` long

The ID of the UserHaiku entity to check for existence.

Returns

Task<bool>

A task that represents the asynchronous operation. The task result contains a boolean value indicating whether the UserHaiku exists.

Namespace Haiku.API.Repositories.UserProfile Repositories

Classes

UserProfileRepository

Interfaces

IUserProfileRepository

Interface IUserProfileRepository

Namespace: Haiku.API.Repositories.UserProfileRepositories

Assembly: Haiku.API.dll

```
public interface IUserProfileRepository
```

Methods

AddUserProfileAsync(UserProfile)

```
Task<UserProfile> AddUserProfileAsync(UserProfile userProfile)
```

Parameters

`userProfile` UserProfile

Returns

Task<UserProfile>

GetAllUserProfilesByUserIdsAsync(List<long>)

```
Task<IEnumerable<UserProfile>> GetAllUserProfilesByUserIdsAsync(List<long> userIds)
```

Parameters

`userIds` List<long>

Returns

Task<IEnumerable<UserProfile>>

GetTotalUserProfilesAsync()

```
Task<int> GetTotalUserProfilesAsync()
```

Returns

Task<int>

GetUserProfileByIdAsync(long)

```
Task<UserProfile?> GetUserProfileByIdAsync(long userProfileId)
```

Parameters

`userProfileId` long

Returns

Task<UserProfile>

GetUserProfileByUserIdAsync(long)

```
Task<UserProfile?> GetUserProfileByUserIdAsync(long userId)
```

Parameters

`userId` long

Returns

Task<UserProfile>

UpdateUserProfileAsync(UserProfile)

```
Task<int> UpdateUserProfileAsync(UserProfile userProfile)
```

Parameters

`userProfile` UserProfile

Returns

Task<int>

UserProfileExistsByIdAsync(long)

```
Task<bool> UserProfileExistsByIdAsync(long userProfileId)
```

Parameters

`userProfileId` long

Returns

Task<bool>

Class UserProfileRepository

Namespace: Haiku.API.Repositories.UserProfileRepositories

Assembly: Haiku.API.dll

```
public class UserProfileRepository : IUserProfileRepository
```

Inheritance

object ← UserProfileRepository

Implements

IUserProfileRepository

Inherited Members

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

Constructors

UserProfileRepository(HaikuAPIContext)

```
public UserProfileRepository(HaikuAPIContext context)
```

Parameters

`context` HaikuAPIContext

Methods

AddUserProfileAsync(UserProfile)

Adds a new UserProfile to the database asynchronously.

```
public Task<UserProfile> AddUserProfileAsync(UserProfile newUserProfile)
```

Parameters

`newUserProfile` UserProfile

The UserProfile entity to be added.

Returns

Task<UserProfile>

A task that represents the asynchronous operation. The task result contains the added UserProfile entity.

GetAllUserProfilesByUserIdsAsync(List<long>)

Retrieves all user profiles that match the provided list of user IDs asynchronously.

```
public Task<IEnumerable<UserProfile>> GetAllUserProfilesByUserIdsAsync(List<long> userIds)
```

Parameters

`userIds` List<long>

A list of user IDs for which the user profiles are to be fetched.

Returns

Task<IEnumerable<UserProfile>>

A task that represents the asynchronous operation. The task result contains an IEnumerable<T> representing the list of user profiles matching the provided user IDs.

GetTotalUserProfilesAsync()

Retrieves the total count of UserProfile entities in the database,

```
public Task<int> GetTotalUserProfilesAsync()
```

Returns

Task<int>

A task that represents the asynchronous operation. The task result contains the total count of Author entities.

GetUserProfileByIdAsync(long)

Retrieves an UserProfile entity by its unique identifier.

```
public Task<UserProfile?> GetUserProfileByIdAsync(long userProfileId)
```

Parameters

`userProfileId` long

The unique identifier of the UserProfile to retrieve.

Returns

Task<UserProfile>

A task that represents the asynchronous operation. The task result contains the UserProfile entity if found; otherwise, `null`.

GetUserProfileByUserIdAsync(long)

Retrieves an UserProfile entity by its User unique identifier.

```
public Task<UserProfile?> GetUserProfileByUserIdAsync(long userId)
```

Parameters

`userId` long

The unique identifier of the UserProfileUser to retrieve.

Returns

Task<UserProfile>

A task that represents the asynchronous operation. The task result contains the UserProfile entity if found; otherwise, `null`.

UpdateUserProfileAsync(UserProfile)

Updates an existing UserProfile entry.

```
public Task<int> UpdateUserProfileAsync(UserProfile updatedUserProfile)
```

Parameters

`updatedUserProfile` UserProfile

The UserProfile entry to update. Must not be null.

Returns

Task<int>

Integer result indicating number of rows affected, if UserProfile was updated.

UserProfileExistsByIdAsync(long)

Checks if an UserProfile entity exists by its ID asynchronously.

```
public Task<bool> UserProfileExistsByIdAsync(long userProfileId)
```

Parameters

`userProfileId` long

The ID of the UserProfile entity to check for existence.

Returns

Task<bool>

A task that represents the asynchronous operation. The task result contains a boolean value indicating whether the UserProfile exists.

Namespace Haiku.API.Repositories.UserRepositories

Classes

UserRepository

Interfaces

IUserRepository

Interface IUserRepository

Namespace: Haiku.API.Repositories.UserRepositories

Assembly: Haiku.API.dll

```
public interface IUserRepository
```

Methods

AddUserAsync(User)

```
Task<User> AddUserAsync(User newUser)
```

Parameters

`newUser` User

Returns

Task<User>

DeleteUserByIdAsync(long)

```
Task<int> DeleteUserByIdAsync(long userId)
```

Parameters

`userId` long

Returns

Task<int>

GetPaginatedUsersAsync(int, int, string)

```
Task<IEnumerable<User>> GetPaginatedUsersAsync(int pageNumber, int pageSize,
string searchOption)
```

Parameters

`pageNumber` int

`pageSize` int

`searchOption` string

Returns

Task<IEnumerable<User>>

GetTotalUsersAsync(string)

```
Task<int> GetTotalUsersAsync(string searchOption)
```

Parameters

`searchOption` string

Returns

Task<int>

GetUserByIdAsync(long)

```
Task<User?> GetUserByIdAsync(long userId)
```

Parameters

`userId` long

Returns

Task<User>

GetUserByUsernameAsync(string)

```
Task<User?> GetUserByUsernameAsync(string username)
```

Parameters

`username` string

Returns

Task<User>

UpdateUserAsync(User)

```
Task<int> UpdateUserAsync(User updatedUser)
```

Parameters

`updatedUser` User

Returns

Task<int>

UserExistsByIdAsync(long)

```
Task<bool> UserExistsByIdAsync(long userId)
```

Parameters

`userId` long

Returns

Task<bool>

UsernameExistsAsync(string)

```
Task<bool> UsernameExistsAsync(string username)
```

Parameters

`username` string

Returns

Task<bool>

Class UserRepository

Namespace: Haiku.API.Repositories.UserRepositories

Assembly: Haiku.API.dll

```
public class UserRepository : IUserRepository
```

Inheritance

object ← UserRepository

Implements

IUserRepository

Inherited Members

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

Constructors

UserRepository(HaikuAPIContext)

```
public UserRepository(HaikuAPIContext context)
```

Parameters

`context` HaikuAPIContext

Methods

AddUserAsync(User)

Adds a new User to the database asynchronously.

```
public Task<User> AddUserAsync(User newUser)
```

Parameters

`newUser` User

The User entity to be added.

Returns

Task<User>

A task that represents the asynchronous operation. The task result contains the added User entity.

DeleteUserByIdAsync(long)

Deletes an existing User entry.

```
public Task<int> DeleteUserByIdAsync(long userId)
```

Parameters

`userId` long

Returns

Task<int>

Integer result indicating number of rows affected, if User was deleted.

GetPaginatedUsersAsync(int, int, string)

Retrieves a paginated list of User entities.

```
public Task<IEnumerable<User>> GetPaginatedUsersAsync(int pageNumber, int pageSize,
string searchOption)
```

Parameters

`pageNumber` int

The number of the page to retrieve. Must be greater than zero.

`pageSize` int

The number of items per page. Must be greater than zero.

`searchOption` string

An optional search term to filter the User entities by username.

Returns

Task<IEnumerable<User>>

A task that represents the asynchronous operation. The task result contains a collection of User entities.

GetTotalUsersAsync(string)

Retrieves the total count of User entities in the database, optionally filtered by a search term for the title.

```
public Task<int> GetTotalUsersAsync(string searchOption)
```

Parameters

`searchOption` string

An optional search term to filter the User entities by username.

Returns

Task<int>

A task that represents the asynchronous operation. The task result contains the total count of User entities.

GetUserByIdAsync(long)

Retrieves an User entity by its unique identifier.

```
public Task<User?> GetUserByIdAsync(long userId)
```

Parameters

`userId` long

The unique identifier of the User to retrieve.

Returns

Task<User>

A task that represents the asynchronous operation. The task result contains the User entity if found; otherwise, `null`.

GetUserByUsernameAsync(string)

Retrieves an User entity by its unique username.

```
public Task<User?> GetUserByUsernameAsync(string username)
```

Parameters

`username` string

The unique username of the User to retrieve.

Returns

Task<User>

A task that represents the asynchronous operation. The task result contains the User entity if found; otherwise, `null`.

UpdateUserAsync(User)

Updates an existing User entry.

```
public Task<int> UpdateUserAsync(User updatedUser)
```

Parameters

`updatedUser` User

The User entry to update. Must not be null.

Returns

Task<int>

Integer result indicating number of rows affected, if User was updated.

UserExistsByIdAsync(long)

Checks if an User entity exists by its ID asynchronously.

```
public Task<bool> UserExistsByIdAsync(long userId)
```

Parameters

`userId` long

The ID of the User entity to check for existence.

Returns

Task<bool>

A task that represents the asynchronous operation. The task result contains a boolean value indicating whether the User exists.

UsernameExistsAsync(string)

Checks if an User entity exists by its unique username.

```
public Task<bool> UsernameExistsAsync(string username)
```

Parameters

`username` string

The unique username of the User to check for existence.

Returns

Task<bool>

A task that represents the asynchronous operation. The task result contains a boolean value indicating whether the User exists.

Namespace Haiku.API.Services.AuthServices

Classes

AuthService

Interfaces

IAuthService

Class AuthService

Namespace: Haiku.API.Services.AuthServices

Assembly: Haiku.API.dll

```
public class AuthService : IAuthService
```

Inheritance

object ← AuthService

Implements

IAuthService

Inherited Members

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

Constructors

AuthService(IUserService, IRoleService, IConfiguration, IMapper, IOptions<JwtSettings>)

```
public AuthService(IUserService userService, IRoleService roleService, IConfiguration
configuration, IMapper mapper, IOptions<JwtSettings> jwtSettings)
```

Parameters

`userService` IUserService

`roleService` IRoleService

`configuration` IConfiguration

`mapper` IMapper

`jwtSettings` IOptions<JwtSettings>

Methods

AuthenticateUserAsync(LoginDto)

Authenticates a User and generates a JSON Web Token (JWT).

```
public Task<JWTokenDto> AuthenticateUserAsync(LoginDto loginDto)
```

Parameters

`loginDto` LoginDto

The LoginDto credentials provided by the User.

Returns

Task <JWTokenDto>

A task that represents the asynchronous operation, containing a JWT token if authentication is successful.

Interface IAuthService

Namespace: Haiku.API.Services.AuthServices

Assembly: Haiku.API.dll

```
public interface IAuthService
```

Methods

AuthenticateUserAsync(LoginDto)

```
Task<JWTokenDto> AuthenticateUserAsync(LoginDto loginDto)
```

Parameters

`loginDto` LoginDto

Returns

Task<JWTokenDto>

Namespace Haiku.API.Services.AuthorHaikuServices

Classes

AuthorHaikuService

Interfaces

IAuthorHaikuService

Class AuthorHaikuService

Namespace: Haiku.API.Services.AuthorHaikuServices

Assembly: Haiku.API.dll

```
public class AuthorHaikuService : IAuthorHaikuService
```

Inheritance

object ← AuthorHaikuService

Implements

IAuthorHaikuService

Inherited Members

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

Constructors

AuthorHaikuService(IAuthorHaikuRepository, IAuthorService, ILogger<AuthorHaikuService>, IMapper)

```
public AuthorHaikuService(IAuthorHaikuRepository authorHaikuRepository, IAuthorService
authorService, ILogger<AuthorHaikuService> logger, IMapper mapper)
```

Parameters

`authorHaikuRepository` IAuthorHaikuRepository

`authorService` IAuthorService

`logger` ILogger<AuthorHaikuService>

`mapper` IMapper

Methods

AddAuthorHaikuAsync(AuthorHaikuDto)

Adds a new AuthorHaiku to the repository.

```
public Task<AuthorHaikuDto> AddAuthorHaikuAsync(AuthorHaikuDto newAuthorHaikuDto)
```

Parameters

`newAuthorHaikuDto` AuthorHaikuDto

The DTO containing the details of the AuthorHaiku to add.

Returns

Task<AuthorHaikuDto>

A task that represents the asynchronous operation. The task result contains the created AuthorHaikuDto.

Exceptions

NotFoundException

Thrown when the Author with the specified ID is not found.

NotSavedException

Thrown when the AuthorHaiku could not be add.

AuthorHaikuExistsByIdAsync(long)

Checks if an AuthorHaiku exists in the repository by its ID.

```
public Task<bool> AuthorHaikuExistsByIdAsync(long authorHaikuId)
```

Parameters

`authorHaikuId` long

The ID of the AuthorHaiku to check.

Returns

Task<bool>

A task that represents the asynchronous operation. The task result contains a boolean value indicating whether the AuthorHaiku exists.

DeleteAuthorHaikuByIdAsync(long)

Deletes an AuthorHaiku from the repository by its ID.

```
public Task DeleteAuthorHaikuByIdAsync(long authorHaikuId)
```

Parameters

`authorHaikuId` long

The ID of the AuthorHaiku to delete.

Returns

Task

A task that represents the asynchronous operation.

Exceptions

NotFoundException

Thrown when the AuthorHaiku with the specified ID is not found.

NotSavedException

Thrown when the AuthorHaiku could not be deleted.

GetAuthorHaikuByIdAsync(long)

Retrieves an AuthorHaikuDto by its unique identifier.

```
public Task<AuthorHaikuDto> GetAuthorHaikuByIdAsync(long authorHaikuId)
```

Parameters

`authorHaikuId` long

The unique identifier of the AuthorHaiku to retrieve.

Returns

Task<AuthorHaikuDto>

A task that represents the asynchronous operation. The task result contains the corresponding AuthorHaikuDto.

Exceptions

NotFoundException

Thrown when the AuthorHaiku with the specified ID is not found.

GetPaginatedAuthorHaikusAsync(int, int, string)

Retrieves a paginated list of AuthorHaikuDto's based on the page number, page size, and optional search criteria.

```
public Task<IEnumerable<AuthorHaikuDto>> GetPaginatedAuthorHaikusAsync(int pageNumber, int pageSize, string searchOption)
```

Parameters

`pageNumber` int

The current page number. Defaults to 1 if a value less than 1 is provided.

`pageSize` int

The number of items per page.

`searchOption` string

An optional search string to filter AuthorHaiku's by their title.

Returns

Task<IEnumerable<AuthorHaikuDto>>

A task that represents the asynchronous operation. The task result contains an IEnumerable<T> of paginated AuthorHaiku's.

GetPaginatedAuthorHaikusByAuthorIdAsync(long, int, int, string)

Retrieves a paginated list of AuthorHaikuDto's based on the Author's ID, page number, page size, and optional search criteria.

```
public Task<IEnumerable<AuthorHaikuDto>> GetPaginatedAuthorHaikusByAuthorIdAsync(long
authorId, int pageNumber, int pageSize, string searchOption)
```

Parameters

`authorId` long

The ID of the Author whose AuthorHaiku's are being retrieved.

`pageNumber` int

The current page number. Defaults to 1 if a value less than 1 is provided.

`pageSize` int

The number of items per page.

`searchOption` string

An optional search string to filter AuthorHaiku's by their title.

Returns

Task<IEnumerable<AuthorHaikuDto>>

A task that represents the asynchronous operation. The task result contains an IEnumerable<T> of paginated AuthorHaiku's by the specified Author.

GetTotalAuthorHaikusAsync(string)

Retrieves the total count of all AuthorHaiku's, optionally filtered by a search string.

```
public Task<int> GetTotalAuthorHaikusAsync(string searchOption)
```

Parameters

`searchOption` string

An optional search string to filter AuthorHaiku's by their title.

Returns

Task <int>

A task that represents the asynchronous operation. The task result contains the total number of AuthorHaiku's.

GetTotalAuthorHaikusByAuthorIdAsync(long, string)

Retrieves the total count of AuthorHaiku's associated with a specific Author, filtered by an optional search criteria.

```
public Task<int> GetTotalAuthorHaikusByAuthorIdAsync(long authorId, string searchOption)
```

Parameters

`authorId` long

The ID of the Author whose AuthorHaiku's are being counted.

`searchOption` string

An optional search string to filter AuthorHaiku's by their title.

Returns

Task <int>

A task that represents the asynchronous operation. The task result contains the total number of AuthorHaiku's by the specified Author.

UpdateAuthorHaikuAsync(long, AuthorHaikuDto)

Updates an existing AuthorHaiku in the repository.

```
public Task UpdateAuthorHaikuAsync(long authorHaikuId, AuthorHaikuDto updatedAuthorHaikuDto)
```

Parameters

`authorHaikuId` long

The ID of the AuthorHaiku to update.

`updatedAuthorHaikuDto` AuthorHaikuDto

The DTO containing the updated details of the AuthorHaiku.

Returns

Task

A task that represents the asynchronous operation.

Exceptions

NotFoundException

Thrown when the AuthorHaiku with the specified ID is not found.

NotSavedException

Thrown when the AuthorHaiku could not be updated.

Interface IAuthorHaikuService

Namespace: Haiku.API.Services.AuthorHaikuServices

Assembly: Haiku.API.dll

```
public interface IAuthorHaikuService
```

Methods

AddAuthorHaikuAsync(AuthorHaikuDto)

```
Task<AuthorHaikuDto> AddAuthorHaikuAsync(AuthorHaikuDto authorHaiku)
```

Parameters

`authorHaiku` AuthorHaikuDto

Returns

Task<AuthorHaikuDto>

AuthorHaikuExistsByIdAsync(long)

```
Task<bool> AuthorHaikuExistsByIdAsync(long authorHaikuId)
```

Parameters

`authorHaikuId` long

Returns

Task<bool>

DeleteAuthorHaikuByIdAsync(long)

```
Task DeleteAuthorHaikuByIdAsync(long authorHaikuId)
```

Parameters

`authorHaikuId` long

Returns

Task

GetAuthorHaikuByIdAsync(long)

```
Task<AuthorHaikuDto> GetAuthorHaikuByIdAsync(long authorHaikuId)
```

Parameters

`authorHaikuId` long

Returns

Task<AuthorHaikuDto>

GetPaginatedAuthorHaikusAsync(int, int, string)

```
Task<IEnumerable<AuthorHaikuDto>> GetPaginatedAuthorHaikusAsync(int pageNumber, int
pageSize, string searchOption)
```

Parameters

`pageNumber` int

`pageSize` int

`searchOption` string

Returns

Task<IEnumerable<AuthorHaikuDto>>

GetPaginatedAuthorHaikusByAuthorIdAsync(long, int, int, string)

```
Task<IEnumerable<AuthorHaikuDto>> GetPaginatedAuthorHaikusByAuthorIdAsync(long authorId, int
pageNumber, int pageSize, string searchOption)
```

Parameters

authorId long

pageNumber int

pageSize int

searchOption string

Returns

Task<IEnumerable<AuthorHaikuDto>>

GetTotalAuthorHaikusAsync(string)

```
Task<int> GetTotalAuthorHaikusAsync(string searchOption)
```

Parameters

searchOption string

Returns

Task<int>

GetTotalAuthorHaikusByAuthorIdAsync(long, string)

```
Task<int> GetTotalAuthorHaikusByAuthorIdAsync(long authorId, string searchOption)
```

Parameters

`authorId` long

`searchOption` string

Returns

Task<int>

UpdateAuthorHaikuAsync(long, AuthorHaikuDto)

```
Task UpdateAuthorHaikuAsync(long authorHaikuId, AuthorHaikuDto existingAuthorHaiku)
```

Parameters

`authorHaikuId` long

`existingAuthorHaiku` AuthorHaikuDto

Returns

Task

Namespace Haiku.API.Services.AuthorServices

Classes

AuthorService

Interfaces

IAuthorService

Class AuthorService

Namespace: Haiku.API.Services.AuthorServices

Assembly: Haiku.API.dll

```
public class AuthorService : IAuthorService
```

Inheritance

object ← AuthorService

Implements

IAuthorService

Inherited Members

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

Constructors

AuthorService(IAuthorRepository, IUnitOfWork, IMapper)

```
public AuthorService(IAuthorRepository authorRepository, IUnitOfWork unitOfWork,
IMapper mapper)
```

Parameters

`authorRepository` IAuthorRepository

`unitOfWork` IUnitOfWork

`mapper` IMapper

Methods

AddAuthorAsync(AuthorDto)

Adds a new author to the system.

```
public Task<AuthorDto> AddAuthorAsync(AuthorDto newAuthorDto)
```

Parameters

`newAuthorDto` AuthorDto

The DTO containing the details of the Author to add.

Returns

Task<AuthorDto>

A task that represents the asynchronous operation. The task result contains the corresponding AuthorDto.

Exceptions

NotSavedException

Thrown when the Author cannot be saved successfully.

AuthorExistsByIdAsync(long)

Checks whether an Author exists in the repository by their unique identifier.

```
public Task<bool> AuthorExistsByIdAsync(long authorId)
```

Parameters

`authorId` long

The unique identifier of the Author to check.

Returns

Task<bool>

A task that represents the asynchronous operation, containing a boolean value indicating whether the Author exists.

DeleteAuthorByIdAsync(long)

Deletes an Author by their unique identifier, along with handling their associated AuthorHaiku's.

```
public Task DeleteAuthorByIdAsync(long authorId)
```

Parameters

`authorId` long

The unique identifier of the Author to be deleted.

Returns

Task

A task that represents the asynchronous operation.

Exceptions

NotFoundException

Thrown when the Author with the specified ID is not found. Or Default Author is not found.

NotSavedException

Thrown when not all associated AuthorHaiku's could be updated to the default Author or when Author was not deleted.

GetAuthorByIdAsync(long)

Retrieves an Author by their unique identifier.

```
public Task<AuthorDto> GetAuthorByIdAsync(long authorId)
```

Parameters

`authorId` long

The unique identifier of the Author to retrieve.

Returns

Task<AuthorDto>

A task that represents the asynchronous operation. The task result contains the corresponding AuthorDto.

Exceptions

NotFoundException

Thrown when the Author with the specified ID is not found.

GetPaginatedAuthorsAsync(int, int, string)

Retrieves a paginated list of AuthorDto, optionally filtered by a search term.

```
public Task<IEnumerable<AuthorDto>> GetPaginatedAuthorsAsync(int pageNumber, int pageSize,
string searchOption)
```

Parameters

`pageNumber` int

The page number to retrieve (1-based index).

`pageSize` int

The number of Author to return per page.

`searchOption` string

An optional search term to filter Author by their properties.

Returns

Task<IEnumerable<AuthorDto>>

A task that represents the asynchronous operation. The task result contains an enumerable collection of AuthorDto objects.

GetTotalAuthorsAsync(string)

Retrieves the total count of Author's, optionally filtered by a search term.

```
public Task<int> GetTotalAuthorsAsync(string searchOption)
```

Parameters

`searchOption` string

An optional search term to filter the Author's.

Returns

Task <int>

A task that represents the asynchronous operation. The task result contains the total count of Author's.

UpdateAuthorAsync(long, AuthorDto)

Updates an existing Author's details in the system.

```
public Task UpdateAuthorAsync(long authorId, AuthorDto updatedAuthorDto)
```

Parameters

`authorId` long

The unique identifier of the Author to update.

`updatedAuthorDto` AuthorDto

The DTO containing the updated details of the Author.

Returns

Task

A task that represents the asynchronous operation.

Exceptions

NotFoundException

Thrown when the Author with the specified ID is not found.

NotSavedException

Thrown when the Author's details cannot be updated successfully.

Interface IAuthorService

Namespace: Haiku.API.Services.AuthorServices

Assembly: Haiku.API.dll

```
public interface IAuthorService
```

Methods

AddAuthorAsync(AuthorDto)

```
Task<AuthorDto> AddAuthorAsync(AuthorDto author)
```

Parameters

`author` AuthorDto

Returns

Task<AuthorDto>

AuthorExistsByIdAsync(long)

```
Task<bool> AuthorExistsByIdAsync(long authorId)
```

Parameters

`authorId` long

Returns

Task<bool>

DeleteAuthorByIdAsync(long)

```
Task DeleteAuthorByIdAsync(long authorId)
```

Parameters

`authorId` long

Returns

Task

GetAuthorByIdAsync(long)

```
Task<AuthorDto> GetAuthorByIdAsync(long authorId)
```

Parameters

`authorId` long

Returns

Task<AuthorDto>

GetPaginatedAuthorsAsync(int, int, string)

```
Task<IEnumerable<AuthorDto>> GetPaginatedAuthorsAsync(int pageNumber, int pageSize,
string searchOption)
```

Parameters

`pageNumber` int

`pageSize` int

`searchOption` string

Returns

Task<IEnumerable<AuthorDto>>

GetTotalAuthorsAsync(string)

```
Task<int> GetTotalAuthorsAsync(string searchOption)
```

Parameters

`searchOption` string

Returns

Task<int>

UpdateAuthorAsync(long, AuthorDto)

```
Task UpdateAuthorAsync(long authorId, AuthorDto existingAuthor)
```

Parameters

`authorId` long

`existingAuthor` AuthorDto

Returns

Task

Namespace Haiku.API.Services.IUserProfileServices

Interfaces

IUserProfileService

Interface IUserProfileService

Namespace: Haiku.API.Services.IUserProfileServices

Assembly: Haiku.API.dll

```
public interface IUserProfileService
```

Methods

AddUserProfileAsync(UserProfileDto)

```
Task<UserProfileDto> AddUserProfileAsync(UserProfileDto userProfile)
```

Parameters

`userProfile` UserProfileDto

Returns

Task<UserProfileDto>

GetAllUserProfilesByUserIdsAsync(List<long>)

```
Task<IEnumerable<UserProfileDto>> GetAllUserProfilesByUserIdsAsync(List<long> userIds)
```

Parameters

`userIds` List<long>

Returns

Task<IEnumerable<UserProfileDto>>

GetTotalUserProfilesAsync()

```
Task<int> GetTotalUserProfilesAsync()
```

Returns

Task<int>

GetUserProfileByIdAsync(long)

```
Task<UserProfileDto> GetUserProfileByIdAsync(long userProfileId)
```

Parameters

`userProfileId` long

Returns

Task<UserProfileDto>

GetUserProfileByUserIdAsync(long)

```
Task<UserProfileDto> GetUserProfileByUserIdAsync(long userId)
```

Parameters

`userId` long

Returns

Task<UserProfileDto>

SaveProfileImageAsync(IFormFile)

```
Task<string> SaveProfileImageAsync(IFormFile file)
```

Parameters

`file` IFormFile

Returns

Task<string>

UpdateUserProfileAsync(long, UserProfileDto)

```
Task UpdateUserProfileAsync(long userProfileId, UserProfileDto existingUserProfile)
```

Parameters

`userProfileId` long

`existingUserProfile` UserProfileDto

Returns

Task

UserProfileExistsByIdAsync(long)

```
Task<bool> UserProfileExistsByIdAsync(long userProfileId)
```

Parameters

`userProfileId` long

Returns

Task<bool>

Namespace Haiku.API.Services.Pagination Service

Classes

PaginationService

Interfaces

IPaginationService

Interface IPaginationService

Namespace: Haiku.API.Services.PaginationService

Assembly: Haiku.API.dll

```
public interface IPaginationService
```

Methods

GetPaginationMetaData(int, int, int)

```
PaginationMetaDataDto GetPaginationMetaData(int totalLogs, int pageSize, int pageNumber)
```

Parameters

`totalLogs` int

`pageSize` int

`pageNumber` int

Returns

PaginationMetaDataDto

Class PaginationService

Namespace: Haiku.API.Services.PaginationService

Assembly: Haiku.API.dll

```
public class PaginationService : IPaginationService
```

Inheritance

object ← PaginationService

Implements

IPaginationService

Inherited Members

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

Methods

GetPaginationMetaData(int, int, int)

Calculates pagination metadata for a collection of logs.

```
public PaginationMetaDataDto GetPaginationMetaData(int totalLogs, int pageSize,
int pageNumber)
```

Parameters

`totalLogs` int

The total number of logs available.

`pageSize` int

The number of logs to display per page.

`pageNumber` int

The current page number.

Returns

PaginationMetaDataDto

A PaginationMetaDataDto object containing pagination information.

Namespace Haiku.API.Services.RoleServices

Classes

RoleService

Interfaces

IRoleService

Interface IRoleService

Namespace: Haiku.API.Services.RoleServices

Assembly: Haiku.API.dll

```
public interface IRoleService
```

Methods

GetUserRoleAsync(long)

```
Task<Role> GetUserRoleAsync(long userId)
```

Parameters

`userId` long

Returns

Task<Role>

Class RoleService

Namespace: Haiku.API.Services.RoleServices

Assembly: Haiku.API.dll

```
public class RoleService : IRoleService
```

Inheritance

object ← RoleService

Implements

IRoleService

Inherited Members

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

Constructors

RoleService(IRoleRepository)

```
public RoleService(IRoleRepository roleRepository)
```

Parameters

`roleRepository` IRoleRepository

Methods

GetUserRoleAsync(long)

Retrieves the Role associated with a specific User.

```
public Task<Role> GetUserRoleAsync(long userId)
```

Parameters

`userId` long

The unique identifier of the User whose Role is to be retrieved.

Returns

Task<Role>

A Role object representing the User's Role.

Exceptions

UnauthorizedAccessException

Thrown when the User's Role cannot be found or retrieved.

Namespace Haiku.API.Services.UserHaiku Services

Classes

UserHaikuService

Interfaces

IUserHaikuService

Interface IUserHaikuService

Namespace: Haiku.API.Services.UserHaikuServices

Assembly: Haiku.API.dll

```
public interface IUserHaikuService
```

Methods

AddUserHaikuAsync(UserHaikuDto)

```
Task<UserHaikuDto> AddUserHaikuAsync(UserHaikuDto userHaiku)
```

Parameters

`userHaiku` UserHaikuDto

Returns

Task<UserHaikuDto>

DeleteUserHaikuByIdAsync(long)

```
Task DeleteUserHaikuByIdAsync(long userHaikuId)
```

Parameters

`userHaikuId` long

Returns

Task

GetPaginatedUserHaikusAsync(int, int, string)

```
Task<IEnumerable<UserHaikuDto>> GetPaginatedUserHaikusAsync(int pageNumber, int pageSize,
string searchOption)
```

Parameters

`pageNumber` int

`pageSize` int

`searchOption` string

Returns

Task<IEnumerable<UserHaikuDto>>

GetPaginatedUserHaikusByUserIdAsync(long, int, int, string)

```
Task<IEnumerable<UserHaikuDto>> GetPaginatedUserHaikusByUserIdAsync(long userId, int
pageNumber, int pageSize, string searchOption)
```

Parameters

`userId` long

`pageNumber` int

`pageSize` int

`searchOption` string

Returns

Task<IEnumerable<UserHaikuDto>>

GetTotalUserHaikusAsync(string)

```
Task<int> GetTotalUserHaikusAsync(string searchOption)
```

Parameters

`searchOption` string

Returns

Task<int>

GetTotalUserHaikusByUserIdAsync(long, string)

```
Task<int> GetTotalUserHaikusByUserIdAsync(long userId, string searchOption)
```

Parameters

`userId` long

`searchOption` string

Returns

Task<int>

GetUserHaikuByIdAsync(long)

```
Task<UserHaikuDto> GetUserHaikuByIdAsync(long userHaikuId)
```

Parameters

`userHaikuId` long

Returns

Task<UserHaikuDto>

UpdateUserHaikuAsync(long, UserHaikuDto)

```
Task UpdateUserHaikuAsync(long userHaikuId, UserHaikuDto existingUserHaiku)
```

Parameters

`userHaikuId` long

`existingUserHaiku` UserHaikuDto

Returns

Task

UserHaikuExistsByIdAsync(long)

```
Task<bool> UserHaikuExistsByIdAsync(long userHaikuId)
```

Parameters

`userHaikuId` long

Returns

Task<bool>

Class UserHaikuService

Namespace: Haiku.API.Services.UserHaikuServices

Assembly: Haiku.API.dll

```
public class UserHaikuService : IUserHaikuService
```

Inheritance

object ← UserHaikuService

Implements

IUserHaikuService

Inherited Members

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

Constructors

UserHaikuService(IUserHaikuRepository, IUserService, ILogger<UserHaikuService>, IMapper)

```
public UserHaikuService(IUserHaikuRepository userHaikuRepository, IUserService userService,
ILogger<UserHaikuService> logger, IMapper mapper)
```

Parameters

`userHaikuRepository` IUserHaikuRepository

`userService` IUserService

`logger` ILogger<UserHaikuService>

`mapper` IMapper

Methods

AddUserHaikuAsync(UserHaikuDto)

Adds a new UserHaiku to the repository.

```
public Task<UserHaikuDto> AddUserHaikuAsync(UserHaikuDto newUserHaikuDto)
```

Parameters

`newUserHaikuDto` UserHaikuDto

The DTO containing the details of the UserHaiku to add.

Returns

Task<UserHaikuDto>

A task that represents the asynchronous operation. The task result contains the created UserHaikuDto.

Exceptions

NotFoundException

Thrown when the User with the specified ID is not found.

NotSavedException

Thrown when the UserHaiku could be updated.

DeleteUserHaikuByIdAsync(long)

Deletes an UserHaiku from the repository by its ID.

```
public Task DeleteUserHaikuByIdAsync(long userHaikuId)
```

Parameters

`userHaikuId` long

The ID of the UserHaiku to delete.

Returns

Task

A task that represents the asynchronous operation.

Exceptions

NotFoundException

Thrown when the UserHaiku with the specified ID is not found.

NotSavedException

Thrown when the UserHaiku could not be deleted.

GetPaginatedUserHaikusAsync(int, int, string)

Retrieves a paginated list of UserHaikuDto's based on the page number, page size, and optional search criteria.

```
public Task<IEnumerable<UserHaikuDto>> GetPaginatedUserHaikusAsync(int pageNumber, int
pageSize, string searchOption)
```

Parameters

`pageNumber` int

The current page number. Defaults to 1 if a value less than 1 is provided.

`pageSize` int

The number of items per page.

`searchOption` string

An optional search string to filter UserHaiku's by their title.

Returns

Task<IEnumerable<UserHaikuDto>>

A task that represents the asynchronous operation. The task result contains an IEnumerable<T> of paginated UserHaiku's.

GetPaginatedUserHaikusByUserIdAsync(long, int, int, string)

Retrieves a paginated list of UserHaikuDto's based on the User's ID, page number, page size, and optional search criteria.

```
public Task<IEnumerable<UserHaikuDto>> GetPaginatedUserHaikusByUserIdAsync(long userId, int pageNumber, int pageSize, string searchOption)
```

Parameters

`userId` long

The ID of the User whose UserHaiku's are being retrieved.

`pageNumber` int

The current page number. Defaults to 1 if a value less than 1 is provided.

`pageSize` int

The number of items per page.

`searchOption` string

An optional search string to filter UserHaiku's by their title.

Returns

Task<IEnumerable<UserHaikuDto>>

A task that represents the asynchronous operation. The task result contains an IEnumerable<T> of paginated UserHaiku's by the specified User.

GetTotalUserHaikusAsync(string)

Retrieves the total count of all UserHaiku's, optionally filtered by a search string.

```
public Task<int> GetTotalUserHaikusAsync(string searchOption)
```

Parameters

`searchOption` string

An optional search string to filter UserHaiku's by their title.

Returns

Task<int>

A task that represents the asynchronous operation. The task result contains the total number of User Haiku's.

GetTotalUserHaikusByUserIdAsync(long, string)

Retrieves the total count of UserHaiku's associated with a specific Author, filtered by an optional search criteria.

```
public Task<int> GetTotalUserHaikusByUserIdAsync(long userId, string searchOption)
```

Parameters

`userId` long

The ID of the User whose UserHaiku's are being counted.

`searchOption` string

An optional search string to filter UserHaiku's by their title.

Returns

Task<int>

A task that represents the asynchronous operation. The task result contains the total number of User Haiku's by the specified User.

GetUserHaikuByIdAsync(long)

Retrieves an UserHaikuDto by its unique identifier.

```
public Task<UserHaikuDto> GetUserHaikuByIdAsync(long userHaikuId)
```

Parameters

`userHaikuId` long

The unique identifier of the UserHaiku to retrieve.

Returns

Task<UserHaikuDto>

A task that represents the asynchronous operation. The task result contains the corresponding UserHaikuDto.

Exceptions

NotFoundException

Thrown when the UserHaiku with the specified ID is not found.

UpdateUserHaikuAsync(long, UserHaikuDto)

Updates an existing UserHaiku in the repository.

```
public Task UpdateUserHaikuAsync(long userHaikuId, UserHaikuDto updatedUserHaikuDto)
```

Parameters

`userHaikuId` long

The ID of the UserHaiku to update.

`updatedUserHaikuDto` UserHaikuDto

The DTO containing the updated details of the UserHaiku.

Returns

Task

A task that represents the asynchronous operation.

Exceptions

NotFoundException

Thrown when the UserHaiku with the specified ID is not found.

NotSavedException

Thrown when the UserHaiku could not be updated.

UserHaikuExistsByIdAsync(long)

Checks if an UserHaiku exists in the repository by its ID.

```
public Task<bool> UserHaikuExistsByIdAsync(long userHaikuId)
```

Parameters

`userHaikuId` long

The ID of the UserHaiku to check.

Returns

Task<bool>

A task that represents the asynchronous operation. The task result contains a boolean value indicating whether the UserHaiku exists.

Namespace Haiku.API.Services.UserProfileServices

Classes

UserProfileService

Class UserProfileService

Namespace: Haiku.API.Services.UserProfileServices

Assembly: Haiku.API.dll

```
public class UserProfileService : IUserProfileService
```

Inheritance

object ← UserProfileService

Implements

IUserProfileService

Inherited Members

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

Constructors

UserProfileService(IUserProfileRepository, IMapper)

```
public UserProfileService(IUserProfileRepository userProfileRepository, IMapper mapper)
```

Parameters

`userProfileRepository` IUserProfileRepository

`mapper` IMapper

Methods

AddUserProfileAsync(UserProfileDto)

Adds a new UserProfile to the repository.

```
public Task<UserProfileDto> AddUserProfileAsync(UserProfileDto newUserProfileDto)
```

Parameters

`newUserProfileDto` UserProfileDto

The DTO containing the details of the UserProfile to add.

Returns

Task<UserProfileDto>

A task that represents the asynchronous operation. The task result contains the created UserProfileDto.

Exceptions

NotSavedException

Thrown when the UserProfile could not be add.

GetAllUserProfilesByUserIdsAsync(List<long>)

Retrieves a paginated list of UserProfileDto's based on the page number, page size, and optional search criteria.

```
public Task<IEnumerable<UserProfileDto>> GetAllUserProfilesByUserIdsAsync(List<long>
userIds)
```

Parameters

`userIds` List<long>

Returns

Task<IEnumerable<UserProfileDto>>

A task that represents the asynchronous operation. The task result contains an IEnumerable<T> of paginated UserHaiku's.

GetTotalUserProfilesAsync()

Retrieves the total count of all UserProfile's, optionally filtered by a search string.

```
public Task<int> GetTotalUserProfilesAsync()
```

Returns

Task<int>

A task that represents the asynchronous operation. The task result contains the total number of User Profile's.

GetUserProfileByIdAsync(long)

Retrieves an UserProfileDto by its unique identifier.

```
public Task<UserProfileDto> GetUserProfileByIdAsync(long userProfileId)
```

Parameters

`userProfileId` long

The unique identifier of the UserProfile to retrieve.

Returns

Task<UserProfileDto>

A task that represents the asynchronous operation. The task result contains the corresponding User ProfileDto.

Exceptions

NotFoundException

Thrown when the UserProfile with the specified ID is not found.

GetUserProfileByUserIdAsync(long)

Retrieves an UserProfileDto by its unique User identifier.

```
public Task<UserProfileDto> GetUserProfileByUserIdAsync(long userId)
```

Parameters

`userId` long

The unique User identifier of the UserProfile to retrieve.

Returns

Task<UserProfileDto>

A task that represents the asynchronous operation. The task result contains the corresponding UserProfileDto.

Exceptions

NotFoundException

Thrown when the UserProfile with the specified ID is not found.

SaveProfileImageAsync(IFormFile)

Saves a profile image to the server's file system.

```
public Task<string> SaveProfileImageAsync(IFormFile file)
```

Parameters

`file` IFormFile

The IFormFile representing the image file to be saved.

Returns

Task<string>

A string representing the relative path to the saved image.

UpdateUserProfileAsync(long, UserProfileDto)

Updates an existing UserProfile in the repository.

```
public Task UpdateUserProfileAsync(long userProfileId, UserProfileDto updatedUserProfileDto)
```

Parameters

`userProfileId` long

The ID of the UserProfile to update.

`updatedUserProfileDto` UserProfileDto

The DTO containing the updated details of the UserProfile.

Returns

Task

A task that represents the asynchronous operation.

Exceptions

NotFoundException

Thrown when the UserProfile with the specified ID is not found.

NotSavedException

Thrown when the UserProfile could not be updated.

UserProfileExistsByIdAsync(long)

Checks if an UserProfile exists in the repository by its ID.

```
public Task<bool> UserProfileExistsByIdAsync(long userProfileId)
```

Parameters

`userProfileId` long

The ID of the UserProfile to check.

Returns

Task<bool>

A task that represents the asynchronous operation. The task result contains a boolean value indicating whether the UserProfile exists.

Namespace Haiku.API.Services.UserServices

Classes

UserService

Interfaces

IUserService

Interface IUserService

Namespace: Haiku.API.Services.UserServices

Assembly: Haiku.API.dll

```
public interface IUserService
```

Methods

AddUserAsync(RegisterDto)

```
Task<UserDto> AddUserAsync(RegisterDto newRegisterDto)
```

Parameters

`newRegisterDto` RegisterDto

Returns

Task<UserDto>

AuthenticateUserAsync(string, string)

```
Task<User> AuthenticateUserAsync(string username, string password)
```

Parameters

`username` string

`password` string

Returns

Task<User>

DeleteUserByIdAsync(long)

```
Task DeleteUserByIdAsync(long userId)
```

Parameters

`userId` long

Returns

Task

GetPaginatedUsersAsync(int, int, string)

```
Task<IEnumerable<UserDto>> GetPaginatedUsersAsync(int pageNumber, int pageSize,
string searchOption)
```

Parameters

`pageNumber` int

`pageSize` int

`searchOption` string

Returns

Task<IEnumerable<UserDto>>

GetTotalUsersAsync(string)

```
Task<int> GetTotalUsersAsync(string searchOption)
```

Parameters

`searchOption` string

Returns

Task<int>

GetUserByIdAsync(long)

```
Task<UserDto> GetUserByIdAsync(long userId)
```

Parameters

`userId` long

Returns

Task<UserDto>

UpdateUserAsync(long, UserDto)

```
Task UpdateUserAsync(long userId, UserDto updatedUserDto)
```

Parameters

`userId` long

`updatedUserDto` UserDto

Returns

Task

UserExistsByIdAsync(long)

```
Task<bool> UserExistsByIdAsync(long userId)
```

Parameters

`userId` long

Returns

Task<bool>

UsernameExistsAsync(string)

```
Task<bool> UsernameExistsAsync(string username)
```

Parameters

`username` string

Returns

Task<bool>

Class UserService

Namespace: Haiku.API.Services.UserServices

Assembly: Haiku.API.dll

```
public class UserService : IUserService
```

Inheritance

object ← UserService

Implements

IUserService

Inherited Members

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

Constructors

UserService(IUserRepository, IUserProfileService, IUnitOfWork, IMapper)

```
public UserService(IUserRepository userRepository, IUserProfileService userprofileService,
IUnitOfWork unitOfWork, IMapper mapper)
```

Parameters

`userRepository` IUserRepository

`userprofileService` IUserProfileService

`unitOfWork` IUnitOfWork

`mapper` IMapper

Methods

AddUserAsync(RegisterDto)

Registers a new User by adding the User's details and creating a UserProfile.

```
public Task<UserDto> AddUserAsync(RegisterDto newRegisterDto)
```

Parameters

`newRegisterDto` RegisterDto

The RegisterDto object containing the User's registration details.

Returns

Task<UserDto>

A UserDto object representing the newly created User.

Exceptions

UsernameAlreadyTakenException

Thrown when the username provided in `newRegisterDto` is already taken.

NotSavedException

Thrown when the User or UserProfile cannot be saved successfully.

Exception

Thrown for any other unexpected errors during the registration process.

AuthenticateUserAsync(string, string)

Authenticates a User based on their username and password.

```
public Task<User> AuthenticateUserAsync(string username, string password)
```

Parameters

`username` string

The username of the User attempting to authenticate.

`password` string

The password provided by the User for authentication.

Returns

Task<User>

A task that represents the asynchronous operation, containing the authenticated User if successful.

DeleteUserByIdAsync(long)

Deletes an User by their unique identifier.

```
public Task DeleteUserByIdAsync(long userId)
```

Parameters

`userId` long

The unique identifier of the User to be deleted.

Returns

Task

A task that represents the asynchronous operation.

Exceptions

NotFoundException

Thrown when the User with the specified ID is not found.

NotSavedException

Thrown when User was not deleted.

GetPaginatedUsersAsync(int, int, string)

Retrieves a paginated list of UserDto, optionally filtered by a search term.

```
public Task<IEnumerable<UserDto>> GetPaginatedUsersAsync(int pageNumber, int pageSize,
string searchOption)
```

Parameters

`pageNumber` int

The page number to retrieve (1-based index).

`pageSize` int

The number of User to return per page.

`searchOption` string

An optional search term to filter User by their properties.

Returns

Task<IEnumerable<UserDto>>

A task that represents the asynchronous operation. The task result contains an enumerable collection of UserDto objects.

GetTotalUsersAsync(string)

Retrieves the total count of User's, optionally filtered by a search term.

```
public Task<int> GetTotalUsersAsync(string searchOption)
```

Parameters

`searchOption` string

An optional search term to filter the User's.

Returns

Task<int>

A task that represents the asynchronous operation. The task result contains the total count of User's.

GetUserByIdAsync(long)

Retrieves an UserDto by their unique identifier.

```
public Task<UserDto> GetUserByIdAsync(long userId)
```

Parameters

`userId` long

The unique identifier of the User to retrieve.

Returns

Task<UserDto>

A task that represents the asynchronous operation. The task result contains the corresponding User Dto.

Exceptions

NotFoundException

Thrown when the User with the specified ID is not found.

UpdateUserAsync(long, UserDto)

Updates an existing User's details in the system.

```
public Task UpdateUserAsync(long userId, UserDto updatedUserDto)
```

Parameters

`userId` long

The unique identifier of the User to update.

`updatedUserDto` UserDto

The DTO containing the updated details of the User.

Returns

Task

A task that represents the asynchronous operation.

Exceptions

NotFoundException

Thrown when the User with the specified ID is not found.

NotSavedException

Thrown when the User's details cannot be updated successfully.

UserExistsByIdAsync(long)

Checks whether an User exists in the repository by their unique identifier.

```
public Task<bool> UserExistsByIdAsync(long userId)
```

Parameters

`userId` long

The unique identifier of the User to check.

Returns

Task<bool>

A task that represents the asynchronous operation, containing a boolean value indicating whether the User exists.

UsernameExistsAsync(string)

Checks whether an User exists in the repository by their unique username.

```
public Task<bool> UsernameExistsAsync(string username)
```

Parameters

`username` string

The unique username of the User to check.

Returns

Task<bool>

A task that represents the asynchronous operation, containing a boolean value indicating whether the User exists.

Namespace Haiku.API.Services.XmlSerialization Services

Interfaces

IXmlSerializationService

Interface IXmlSerializationService

Namespace: Haiku.API.Services.XmlSerializationServices

Assembly: Haiku.API.dll

```
public interface IXmlSerializationService
```

Methods

SerializeAndSanitizeToXml(PaginationMetaDataDto)

```
string SerializeAndSanitizeToXml(PaginationMetaDataDto paginationMetaData)
```

Parameters

`paginationMetaData` PaginationMetaDataDto

Returns

string

Namespace Haiku.API.Utilities

Classes

MinLengthIfNotEmptyAttribute

StringLengthIfNotEmptyAttribute

UnitOfWork

Interfaces

IUnitOfWork

Interface IUnitOfWork

Namespace: Haiku.API.Utilities

Assembly: Haiku.API.dll

```
public interface IUnitOfWork : IDisposable
```

Inherited Members
IDisposable.Dispose()

Properties

AuthorHaikus

```
IAuthorHaikuRepository AuthorHaikus { get; }
```

Property Value

IAuthorHaikuRepository

Authors

```
IAuthorRepository Authors { get; }
```

Property Value

IAuthorRepository

UserProfiles

```
IUserProfileRepository UserProfiles { get; }
```

Property Value

IUserProfileRepository

Users

```
IUserRepository Users { get; }
```

Property Value

IUserRepository

Methods

BeginTransactionAsync()

```
Task BeginTransactionAsync()
```

Returns

Task

CommitAsync()

```
Task CommitAsync()
```

Returns

Task

CompleteAsync()

```
Task<int> CompleteAsync()
```

Returns

Task<int>

RollbackAsync()

```
Task RollbackAsync()
```

Returns

Task

Class MinLengthIfNotEmptyAttribute

Namespace: Haiku.API.Utilities

Assembly: Haiku.API.dll

```
public class MinLengthIfNotEmptyAttribute : ValidationAttribute
```

Inheritance

object ← Attribute ← ValidationAttribute ← MinLengthIfNotEmptyAttribute

Inherited Members

ValidationAttribute.FormatErrorMessage(string) ,
ValidationAttribute.GetValidationResult(object, ValidationContext) ,
ValidationAttribute.IsValid(object) , ValidationAttribute.Validate(object, ValidationContext) ,
ValidationAttribute.Validate(object, string) , ValidationAttribute.ErrorMessage ,
ValidationAttribute.ErrorMessageResourceName , ValidationAttribute.ErrorMessageResourceType ,
ValidationAttribute.ErrorMessageString , ValidationAttribute.RequiresValidationContext ,
Attribute.Equals(object) , Attribute.GetCustomAttribute(Assembly, Type) ,
Attribute.GetCustomAttribute(Assembly, Type, bool) ,
Attribute.GetCustomAttribute(MemberInfo, Type) ,
Attribute.GetCustomAttribute(MemberInfo, Type, bool) ,
Attribute.GetCustomAttribute(Module, Type) , Attribute.GetCustomAttribute(Module, Type, bool) ,
Attribute.GetCustomAttribute(ParameterInfo, Type) ,
Attribute.GetCustomAttribute(ParameterInfo, Type, bool) , Attribute.GetCustomAttributes(Assembly) ,
Attribute.GetCustomAttributes(Assembly, bool) , Attribute.GetCustomAttributes(Assembly, Type) ,
Attribute.GetCustomAttributes(Assembly, Type, bool) , Attribute.GetCustomAttributes(MemberInfo) ,
Attribute.GetCustomAttributes(MemberInfo, bool) ,
Attribute.GetCustomAttributes(MemberInfo, Type) ,
Attribute.GetCustomAttributes(MemberInfo, Type, bool) , Attribute.GetCustomAttributes(Module) ,
Attribute.GetCustomAttributes(Module, bool) , Attribute.GetCustomAttributes(Module, Type) ,
Attribute.GetCustomAttributes(Module, Type, bool) , Attribute.GetCustomAttributes(ParameterInfo) ,
Attribute.GetCustomAttributes(ParameterInfo, bool) ,
Attribute.GetCustomAttributes(ParameterInfo, Type) ,
Attribute.GetCustomAttributes(ParameterInfo, Type, bool) , Attribute.GetHashCode() ,
Attribute.IsDefaultAttribute() , Attribute.IsDefined(Assembly, Type) ,
Attribute.IsDefined(Assembly, Type, bool) , Attribute.IsDefined(MemberInfo, Type) ,
Attribute.IsDefined(MemberInfo, Type, bool) , Attribute.IsDefined(Module, Type) ,
Attribute.IsDefined(Module, Type, bool) , Attribute.IsDefined(ParameterInfo, Type) ,
Attribute.IsDefined(ParameterInfo, Type, bool) , Attribute.Match(object) , Attribute.TypeId ,

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

Constructors

MinLengthIfNotEmptyAttribute(int, string)

Initializes a new instance of the MinLengthIfNotEmptyAttribute class. This attribute enforces a minimum length for a string property, only if the property is not empty.

```
public MinLengthIfNotEmptyAttribute(int minLength, string field)
```

Parameters

`minLength` int

The minimum required length for the string property. Must be a non-negative value.

`field` string

The name of the field being validated. This will be used in the error message.

Methods

IsValid(object?, ValidationContext)

Validates the specified value with respect to the current validation attribute.

```
protected override ValidationResult IsValid(object? value, ValidationContext
validationContext)
```

Parameters

`value` object

The value to validate.

`validationContext` ValidationContext

The context information about the validation operation.

Returns

ValidationResult

An instance of the ValidationResult class.

Exceptions

InvalidOperationException

The current attribute is malformed.

NotImplementedException

IsValid(object, ValidationContext) has not been implemented by a derived class.

Class StringLengthIfNotEmptyAttribute

Namespace: Haiku.API.Utilities

Assembly: Haiku.API.dll

```
public class StringLengthIfNotEmptyAttribute : ValidationAttribute
```

Inheritance

object ← Attribute ← ValidationAttribute ← StringLengthIfNotEmptyAttribute

Inherited Members

ValidationAttribute.FormatErrorMessage(string) ,
ValidationAttribute.GetValidationResult(object, ValidationContext) ,
ValidationAttribute.IsValid(object) , ValidationAttribute.Validate(object, ValidationContext) ,
ValidationAttribute.Validate(object, string) , ValidationAttribute.ErrorMessage ,
ValidationAttribute.ErrorMessageResourceName , ValidationAttribute.ErrorMessageResourceType ,
ValidationAttribute.ErrorMessageString , ValidationAttribute.RequiresValidationContext ,
Attribute.Equals(object) , Attribute.GetCustomAttribute(Assembly, Type) ,
Attribute.GetCustomAttribute(Assembly, Type, bool) ,
Attribute.GetCustomAttribute(MemberInfo, Type) ,
Attribute.GetCustomAttribute(MemberInfo, Type, bool) ,
Attribute.GetCustomAttribute(Module, Type) , Attribute.GetCustomAttribute(Module, Type, bool) ,
Attribute.GetCustomAttribute(ParameterInfo, Type) ,
Attribute.GetCustomAttribute(ParameterInfo, Type, bool) , Attribute.GetCustomAttributes(Assembly) ,
Attribute.GetCustomAttributes(Assembly, bool) , Attribute.GetCustomAttributes(Assembly, Type) ,
Attribute.GetCustomAttributes(Assembly, Type, bool) , Attribute.GetCustomAttributes(MemberInfo) ,
Attribute.GetCustomAttributes(MemberInfo, bool) ,
Attribute.GetCustomAttributes(MemberInfo, Type) ,
Attribute.GetCustomAttributes(MemberInfo, Type, bool) , Attribute.GetCustomAttributes(Module) ,
Attribute.GetCustomAttributes(Module, bool) , Attribute.GetCustomAttributes(Module, Type) ,
Attribute.GetCustomAttributes(Module, Type, bool) , Attribute.GetCustomAttributes(ParameterInfo) ,
Attribute.GetCustomAttributes(ParameterInfo, bool) ,
Attribute.GetCustomAttributes(ParameterInfo, Type) ,
Attribute.GetCustomAttributes(ParameterInfo, Type, bool) , Attribute.GetHashCode() ,
Attribute.IsDefaultAttribute() , Attribute.IsDefined(Assembly, Type) ,
Attribute.IsDefined(Assembly, Type, bool) , Attribute.IsDefined(MemberInfo, Type) ,
Attribute.IsDefined(MemberInfo, Type, bool) , Attribute.IsDefined(Module, Type) ,
Attribute.IsDefined(Module, Type, bool) , Attribute.IsDefined(ParameterInfo, Type) ,
Attribute.IsDefined(ParameterInfo, Type, bool) , Attribute.Match(object) , Attribute.TypeId ,

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

Constructors

StringLengthIfNotEmptyAttribute(int, string)

Initializes a new instance of the MinLengthIfNotEmptyAttribute class. This attribute enforces a maximum length for a string property, only if the property is not empty.

```
public StringLengthIfNotEmptyAttribute(int maxLength, string field)
```

Parameters

`maxLength` int

The maximum required length for the string property. Must be a non-negative value.

`field` string

The name of the field being validated. This will be used in the error message.

Methods

IsValid(object?, ValidationContext)

Validates the specified value with respect to the current validation attribute.

```
protected override ValidationResult IsValid(object? value, ValidationContext
validationContext)
```

Parameters

`value` object

The value to validate.

`validationContext` ValidationContext

The context information about the validation operation.

Returns

ValidationResult

An instance of the ValidationResult class.

Exceptions

InvalidOperationException

The current attribute is malformed.

NotImplementedException

IsValid(object, ValidationContext) has not been implemented by a derived class.

Class UnitOfWork

Namespace: Haiku.API.Utilities

Assembly: Haiku.API.dll

```
public class UnitOfWork : IUnitOfWork, IDisposable
```

Inheritance

object ← UnitOfWork

Implements

IUnitOfWork, IDisposable

Inherited Members

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

Constructors

UnitOfWork(HaikuAPIContext, IAuthorRepository, IAuthorHaikuRepository, IUserRepository, IUserProfileRepository)

Initializes a new instance of the UnitOfWork class.

```
public UnitOfWork(HaikuAPIContext context, IAuthorRepository authors, IAuthorHaikuRepository
authorHaikus, IUserRepository users, IUserProfileRepository userProfiles)
```

Parameters

`context` HaikuAPIContext

The database context to be used for operations.

`authors` IAuthorRepository

The repository for managing authors.

`authorHaikus` IAuthorHaikuRepository

The repository for managing author haikus.

`users` IUserRepository

The repository for managing users.

`userProfiles` IUserProfileRepository

The repository for managing user profiles.

Properties

AuthorHaikus

Gets the repository for managing author haikus.

```
public IAuthorHaikuRepository AuthorHaikus { get; }
```

Property Value

IAuthorHaikuRepository

Authors

Gets the repository for managing authors.

```
public IAuthorRepository Authors { get; }
```

Property Value

IAuthorRepository

UserProfiles

Gets the repository for managing user profiles.

```
public IUserProfileRepository UserProfiles { get; }
```

Property Value

IUserProfileRepository

Users

Gets the repository for managing users.

```
public IUserRepository Users { get; }
```

Property Value

IUserRepository

Methods

BeginTransactionAsync()

Begins a new database transaction asynchronously.

```
public Task BeginTransactionAsync()
```

Returns

Task

A task that represents the asynchronous operation.

Remarks

This method should be called before performing a series of operations that need to be executed as a single unit of work.

CommitAsync()

Commits the current transaction asynchronously.

```
public Task CommitAsync()
```

Returns

Task

A task that represents the asynchronous operation.

Remarks

This method should be called after all operations are complete to persist changes to the database.

CompleteAsync()

Saves all changes made in this unit of work asynchronously.

```
public Task<int> CompleteAsync()
```

Returns

Task <int>

The number of state entries written to the database.

Dispose()

Disposes the resources used by the UnitOfWork class.

```
public void Dispose()
```

RollbackAsync()

Rolls back the current transaction asynchronously.

```
public Task RollbackAsync()
```

Returns

Task

A task that represents the asynchronous operation.

Remarks

This method should be called if an error occurs during the operations to revert all changes.

Namespace Haiku.API.Utility

Classes

SyllableCountAttribute

Class SyllableCountAttribute

Namespace: Haiku.API.Utility

Assembly: Haiku.API.dll

```
[AttributeUsage(AttributeTargets.Property, AllowMultiple = false)]
public class SyllableCountAttribute : ValidationAttribute
```

Inheritance

object ← Attribute ← ValidationAttribute ← SyllableCountAttribute

Inherited Members

ValidationAttribute.FormatErrorMessage(string) ,
ValidationAttribute.GetValidationResult(object, ValidationContext) ,
ValidationAttribute.IsValid(object) , ValidationAttribute.Validate(object, ValidationContext) ,
ValidationAttribute.Validate(object, string) , ValidationAttribute.ErrorMessage ,
ValidationAttribute.ErrorMessageResourceName , ValidationAttribute.ErrorMessageResourceType ,
ValidationAttribute.ErrorMessageString , ValidationAttribute.RequiresValidationContext ,
Attribute.Equals(object) , Attribute.GetCustomAttribute(Assembly, Type) ,
Attribute.GetCustomAttribute(Assembly, Type, bool) ,
Attribute.GetCustomAttribute(MemberInfo, Type) ,
Attribute.GetCustomAttribute(MemberInfo, Type, bool) ,
Attribute.GetCustomAttribute(Module, Type) , Attribute.GetCustomAttribute(Module, Type, bool) ,
Attribute.GetCustomAttribute(ParameterInfo, Type) ,
Attribute.GetCustomAttribute(ParameterInfo, Type, bool) , Attribute.GetCustomAttributes(Assembly) ,
Attribute.GetCustomAttributes(Assembly, bool) , Attribute.GetCustomAttributes(Assembly, Type) ,
Attribute.GetCustomAttributes(Assembly, Type, bool) , Attribute.GetCustomAttributes(MemberInfo) ,
Attribute.GetCustomAttributes(MemberInfo, bool) ,
Attribute.GetCustomAttributes(MemberInfo, Type) ,
Attribute.GetCustomAttributes(MemberInfo, Type, bool) , Attribute.GetCustomAttributes(Module) ,
Attribute.GetCustomAttributes(Module, bool) , Attribute.GetCustomAttributes(Module, Type) ,
Attribute.GetCustomAttributes(Module, Type, bool) , Attribute.GetCustomAttributes(ParameterInfo) ,
Attribute.GetCustomAttributes(ParameterInfo, bool) ,
Attribute.GetCustomAttributes(ParameterInfo, Type) ,
Attribute.GetCustomAttributes(ParameterInfo, Type, bool) , Attribute.GetHashCode() ,
Attribute.IsDefaultAttribute() , Attribute.IsDefined(Assembly, Type) ,
Attribute.IsDefined(Assembly, Type, bool) , Attribute.IsDefined(MemberInfo, Type) ,
Attribute.IsDefined(MemberInfo, Type, bool) , Attribute.IsDefined(Module, Type) ,
Attribute.IsDefined(Module, Type, bool) , Attribute.IsDefined(ParameterInfo, Type) ,

Attribute.IsDefined(ParameterInfo, Type, bool) , Attribute.Match(object) , Attribute.TypeId , object.Equals(object, object) , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

SyllableCountAttribute(int)

Initializes a new instance of the SyllableCountAttribute class.

```
public SyllableCountAttribute(int expectedCount)
```

Parameters

`expectedCount` int

The expected number of syllables for validation purposes.

Remarks

This attribute is used to specify the expected syllable count for a string. It can be applied to properties in model classes to enforce syllable constraints.

Methods

BasicCountSyllables(string)

Counts the total number of syllables in the given line of text.

```
public static int BasicCountSyllables(string line)
```

Parameters

`line` string

The line of text to analyze. If null or empty, returns 0.

Returns

int

The total number of syllables found in the `line`.

Remarks

The method splits the input line into words based on whitespace and common punctuation, then calculates the total syllables by calling BasicCountSyllablesInWord(string) for each word.

IsValid(object?, ValidationContext)

Validates the specified value with respect to the current validation attribute.

```
protected override ValidationResult IsValid(object? value, ValidationContext
validationContext)
```

Parameters

`value` object

The value to validate.

`validationContext` ValidationContext

The context information about the validation operation.

Returns

ValidationResult

An instance of the ValidationResult class.

Exceptions

InvalidOperationException

The current attribute is malformed.

NotImplementedException

IsValid(object, ValidationContext) has not been implemented by a derived class.

Namespace HaikuApi.Tests

Classes

CustomWebApplicationFactory<TProgram>

Class CustomWebApplicationFactory<TProgram>

Namespace: HaikuApi.Tests

Assembly: HaikuApi.Tests.dll

```
public class CustomWebApplicationFactory<TProgram> : WebApplicationFactory<TProgram>,
IDisposable, IAsyncDisposable where TProgram : class
```

Type Parameters

`TProgram`

Inheritance

object ← WebApplicationFactory<TProgram> ← CustomWebApplicationFactory<TProgram>

Implements

IDisposable, IAsyncDisposable

Inherited Members

WebApplicationFactory<TProgram>.WithWebHostBuilder(Action<IWebHostBuilder>) ,
WebApplicationFactory<TProgram>.GetTestAssemblies() ,
WebApplicationFactory<TProgram>.CreateHostBuilder() ,
WebApplicationFactory<TProgram>.CreateWebHostBuilder() ,
WebApplicationFactory<TProgram>.CreateServer(IWebHostBuilder) ,
WebApplicationFactory<TProgram>.CreateHost(IHostBuilder) ,
WebApplicationFactory<TProgram>.CreateClient() ,
WebApplicationFactory<TProgram>.CreateClient(WebApplicationFactoryClientOptions) ,
WebApplicationFactory<TProgram>.CreateDefaultClient(params DelegatingHandler[]) ,
WebApplicationFactory<TProgram>.ConfigureClient(HttpClient) ,
WebApplicationFactory<TProgram>.CreateDefaultClient(Uri, params DelegatingHandler[]) ,
WebApplicationFactory<TProgram>.Dispose() , WebApplicationFactory<TProgram>.Dispose(bool) ,
WebApplicationFactory<TProgram>.DisposeAsync() , WebApplicationFactory<TProgram>.Server ,
WebApplicationFactory<TProgram>.Services , WebApplicationFactory<TProgram>.Factories ,
WebApplicationFactory<TProgram>.ClientOptions , object.Equals(object) ,
object.Equals(object, object) , object.GetHashCode() , object.GetType() ,
object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

ConfigureWebHost(IWebHostBuilder)

Gives a fixture an opportunity to configure the application before it gets built.

```
protected override void ConfigureWebHost(IWebHostBuilder builder)
```

Parameters

`builder` IWebHostBuilder

The IWebHostBuilder for the application.

Namespace HaikuApi.Tests.TestUtilities

Classes

TestAuthHandler

Class TestAuthHandler

Namespace: HaikuApi.Tests.TestUtilities

Assembly: HaikuApi.Tests.dll

```
public class TestAuthHandler : AuthenticationHandler<AuthenticationSchemeOptions>,
IAuthenticationHandler
```

Inheritance

object ← AuthenticationHandler<AuthenticationSchemeOptions> ← TestAuthHandler

Implements

IAuthenticationHandler

Inherited Members

AuthenticationHandler<AuthenticationSchemeOptions>.InitializeAsync(AuthenticationScheme, HttpContext) ,
AuthenticationHandler<AuthenticationSchemeOptions>.InitializeEventsAsync() ,
AuthenticationHandler<AuthenticationSchemeOptions>.CreateEventsAsync() ,
AuthenticationHandler<AuthenticationSchemeOptions>.InitializeHandlerAsync() ,
AuthenticationHandler<AuthenticationSchemeOptions>.BuildRedirectUri(string) ,
AuthenticationHandler<AuthenticationSchemeOptions>.ResolveTarget(string) ,
AuthenticationHandler<AuthenticationSchemeOptions>.AuthenticateAsync() ,
AuthenticationHandler<AuthenticationSchemeOptions>.HandleAuthenticateOnceAsync() ,
AuthenticationHandler<AuthenticationSchemeOptions>.HandleAuthenticateOnceSafeAsync() ,
AuthenticationHandler<AuthenticationSchemeOptions>.HandleForbiddenAsync(AuthenticationProperties) ,
AuthenticationHandler<AuthenticationSchemeOptions>.HandleChallengeAsync(AuthenticationProperties) ,
AuthenticationHandler<AuthenticationSchemeOptions>.ChallengeAsync(AuthenticationProperties) ,
AuthenticationHandler<AuthenticationSchemeOptions>.ForbidAsync(AuthenticationProperties) ,
AuthenticationHandler<AuthenticationSchemeOptions>.Scheme ,
AuthenticationHandler<AuthenticationSchemeOptions>.Options ,
AuthenticationHandler<AuthenticationSchemeOptions>.Context ,
AuthenticationHandler<AuthenticationSchemeOptions>.Request ,
AuthenticationHandler<AuthenticationSchemeOptions>.Response ,
AuthenticationHandler<AuthenticationSchemeOptions>.OriginalPath ,
AuthenticationHandler<AuthenticationSchemeOptions>.OriginalPathBase ,
AuthenticationHandler<AuthenticationSchemeOptions>.Logger ,

AuthenticationHandler<AuthenticationSchemeOptions>.UrlEncoder , AuthenticationHandler<AuthenticationSchemeOptions>.Clock , AuthenticationHandler<AuthenticationSchemeOptions>.TimeProvider , AuthenticationHandler<AuthenticationSchemeOptions>.OptionsMonitor , AuthenticationHandler<AuthenticationSchemeOptions>.Events , AuthenticationHandler<AuthenticationSchemeOptions>.ClaimsIssuer , AuthenticationHandler<AuthenticationSchemeOptions>.CurrentUri , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

TestAuthHandler(IOptionsMonitor<AuthenticationSchemeOptions>, ILoggerFactory, UrlEncoder)

```
public TestAuthHandler(IOptionsMonitor<AuthenticationSchemeOptions> options, ILoggerFactory logger, UrlEncoder encoder)
```

Parameters

`options` IOptionsMonitor<AuthenticationSchemeOptions>

`logger` ILoggerFactory

`encoder` UrlEncoder

Methods

HandleAuthenticateAsync()

Allows derived types to handle authentication.

```
protected override Task<AuthenticateResult> HandleAuthenticateAsync()
```

Returns

Task<AuthenticateResult>

The AuthenticateResult.

Namespace HaikuApi.Tests.UnitTests

Classes

AuthorServiceTests

Class AuthorServiceTests

Namespace: HaikuApi.Tests.UnitTests

Assembly: HaikuApi.Tests.dll

```
public class AuthorServiceTests
```

Inheritance

object ← AuthorServiceTests

Inherited Members

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

Constructors

AuthorServiceTests(ITestOutputHelper)

```
public AuthorServiceTests(ITestOutputHelper output)
```

Parameters

`output` ITestOutputHelper

Methods

CreateAuthorAsync_AddAuthor()

```
[Fact]
public Task CreateAuthorAsync_AddAuthor()
```

Returns

Task

DeleteAuthorAsync_DeleteAuthor()

```
[Fact]
public Task DeleteAuthorAsync_DeleteAuthor()
```

Returns

Task

GetAuthorAsync_ReturnAuthor()

```
[Fact]
public Task GetAuthorAsync_ReturnAuthor()
```

Returns

Task

UpdateAuthorAsync_UpdateAuthor()

```
[Fact]
public Task UpdateAuthorAsync_UpdateAuthor()
```

Returns

Task