Namespace TokenManager.Api.Controllers

Classes

UserController

Class UserController

Namespace: TokenManager.Api.Controllers

Assembly: TokenManager.Api.dll

```
[Route("api/v1")]
[ApiController]
public class UserController : Controller, IActionFilter, IAsyncActionFilter,
IFilterMetadata, IDisposable
```

Inheritance

object ← ControllerBase ← Controller ← UserController

Implements

IActionFilter, IAsyncActionFilter, IFilterMetadata, IDisposable

Inherited Members

Controller.View() , Controller.View(string) , Controller.View(object) ,
Controller.View(string, object) , Controller.PartialView() , Controller.PartialView(string) ,
Controller.PartialView(object) , Controller.PartialView(string, object) ,
Controller.ViewComponent(string) , Controller.ViewComponent(Type) ,
Controller.ViewComponent(string, object) , Controller.ViewComponent(Type, object) ,
Controller.Json(object) , Controller.Json(object, object) ,
Controller.OnActionExecuting(ActionExecutingContext) ,
Controller.OnActionExecuted(ActionExecutedContext) ,
Controller.OnActionExecutionAsync(ActionExecutingContext, ActionExecutionDelegate) ,
Controller.Dispose() , Controller.Dispose(bool) , Controller.ViewData , Controller.TempData ,
Controller.ViewBag , 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(IMediator)

```
public UserController(IMediator mediator)
```

Parameters

`mediator` IMediator

Methods

CreateUser(string, AddUserRequest, CancellationToken)

Add a new user on the keycloak realm.

```
[HttpPost]
[Route("createUser/{tenant}", Name = "CreateUser")]
[ProducesResponseType(201)]
[ProducesResponseType(500)]
[ProducesResponseType(400)]
public Task<IActionResult> CreateUser(string tenant, AddUserRequest addUserRequest,
CancellationToken cancellationToken)
```

Parameters

`tenant` string

`addUserRequest` AddUserRequest

`cancellationToken` CancellationToken

Returns

Task <IActionResult>

A status code related to the operation.

Login(string, LoginUserRequest, CancellationToken)

Return a valid JWT token and details about them.

```
[HttpPost]
[Route("login/{tenant}", Name = "Login")]
[ProducesResponseType(201)]
[ProducesResponseType(500)]
```

```
[ProducesResponseType(400)]
public Task<IActionResult> Login(string tenant, LoginUserRequest loginUserRequest,
CancellationToken cancellationToken)
```

Parameters

`tenant` string

`loginUserRequest` LoginUserRequest

`cancellationToken` CancellationToken

Returns

Task<IActionResult>

A status code related to the operation.

RefreshToken(string, RefreshTokenRequest, CancellationToken)

Return a valid JWT token and details about them refreshed.

```
[HttpPost]
[Route("refreshToken/{tenant}", Name = "RefreshToken")]
[ProducesResponseType(201)]
[ProducesResponseType(500)]
[ProducesResponseType(400)]
public Task<IActionResult> RefreshToken(string tenant, RefreshTokenRequest request,
CancellationToken cancellationToken)
```

Parameters

`tenant` string

`request` RefreshTokenRequest

`cancellationToken` CancellationToken

Returns

Task<IActionResult>

A status code related to the operation.