Inertia/Backend

Generated by Doxygen 1.9.1

Chapter 1

inertia / Backend

This is the directory for the backend project component of the `inertia` software.

This directory contains the the project structure of this component.

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

Class Documentation

4.1 Account Class Reference

Properties

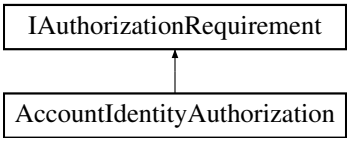
- string **AccountId** = null! `[get, set]`
- string **Name** = null! `[get, set]`
- string **Email** = null! `[get, set]`
- string **Password** = null! `[get, set]`
- string **Salt** = null! `[get, set]`
- AccountRole **Role** `[get, set]`
- AccountState **State** `[get, set]`
- UserType **UserType** `[get, set]`

The documentation for this class was generated from the following file:

- Models/Account.cs

4.2 AccountIdentityAuthorization Class Reference

Inheritance diagram for AccountIdentityAuthorization:



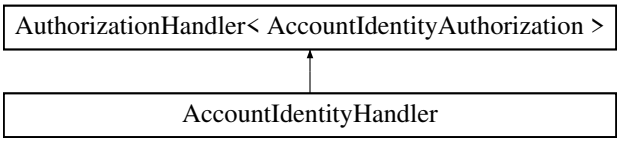
The documentation for this class was generated from the following file:

- Authorization/AccountIdentityAuthorization.cs

4.3 AccountIdentityHandler Class Reference

Handles the MatchAccountId policy - if the access token id matches the one in the request.

Inheritance diagram for AccountIdentityHandler:



Public Member Functions

- **AccountIdentityHandler** (IHttpContextAccessor httpContextAccessor)

Protected Member Functions

- override Task **HandleRequirementAsync** (AuthorizationHandlerContext context, AccountIdentityAuthorization requirement)

4.3.1 Detailed Description

Handles the MatchAccountId policy - if the access token id matches the one in the request.

The documentation for this class was generated from the following file:

- Authorization/AccountIdentityHandler.cs

4.4 AuthenticationTokenService Class Reference

Generates Authentication tokens, to safely and securely authenticate and authorize users.

Public Member Functions

- **AuthenticationTokenService** (IConfiguration configuration)
- string? GenerateAccessToken (Account account)

  Generates an access token for an account
- ClaimsPrincipal? ValidateAccessToken (string token)

  Checks whether a token is valid.

4.4.1 Detailed Description

Generates Authentication tokens, to safely and securely authenticate and authorize users.

4.4.2 Member Function Documentation

4.4.2.1 GenerateAccessToken()

```
string?  AuthenticationTokenService.GenerateAccessToken (
            Account account )  [inline]
```

Generates an access token for an account

Parameters

account	

Returns

4.4.2.2 ValidateAccessToken()

```
ClaimsPrincipal?  AuthenticationTokenService.ValidateAccessToken (
            string token )  [inline]
```

Checks whether a token is valid.

Parameters

token	

Returns

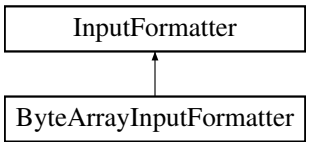
The documentation for this class was generated from the following file:

- Services/AuthenticationTokenService.cs

4.5 ByteArrayInputFormatter Class Reference

Utility class to format application/octet-stream request body into a byte[].

Inheritance diagram for ByteArrayInputFormatter:



Public Member Functions

- override async Task< InputFormatterResult > **ReadRequestBodyAsync** (InputFormatterContext context)

Protected Member Functions

- override bool **CanReadType** (Type type)

4.5.1 Detailed Description

Utility class to format application/octet-stream request body into a byte[].

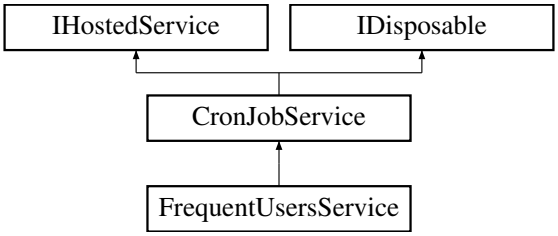
The documentation for this class was generated from the following file:

- Util/ByteArrayInputFormatter.cs

4.6 CronJobService Class Reference

Utility Service to run recurring tasks after a cron expression.

Inheritance diagram for CronJobService:



Public Member Functions

- virtual async Task **StartAsync** (CancellationToken cancellationToken)
- virtual async Task **DoWork** (CancellationToken cancellationToken)
- virtual async Task **StopAsync** (CancellationToken cancellationToken)
- virtual void **Dispose** ()

Protected Member Functions

- **CronJobService** (string cronExpression, TimeZoneInfo timeZoneInfo)
- virtual async Task **ScheduleJob** (CancellationToken cancellationToken)

4.6.1 Detailed Description

Utility Service to run recurring tasks after a cron expression.

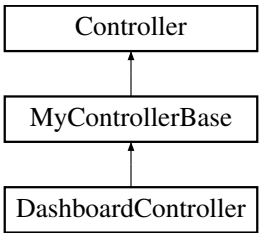
The documentation for this class was generated from the following file:

- Services/CronJobService.cs

4.7 DashboardController Class Reference

The controller for the admin dashboard functionality

Inheritance diagram for DashboardController:



Public Member Functions

- **DashboardController** (InertiaContext db, InertiaService inertia)
- async Task< ActionResult< DashboardResponse > > Get ()

  Fetches all the dashboard data and sends it in the http response.

Additional Inherited Members

4.7.1 Detailed Description

The controller for the admin dashboard functionality

4.7.2 Member Function Documentation

4.7.2.1 Get()

```
async Task<ActionResult<DashboardResponse> > DashboardController.Get ( )  [inline]
```

Fetches all the dashboard data and sends it in the http response.

Returns

The documentation for this class was generated from the following file:

- Controllers/Admin/DashboardController.cs

4.8 DbInitializer Class Reference

Initializes database with the required depos and scooters, as well as the test accounts and orders.

Static Public Member Functions

- static void **Initialize** (InertiaContext context, UsersService users, IWebHostEnvironment env)

4.8.1 Detailed Description

Initializes database with the required depos and scooters, as well as the test accounts and orders.

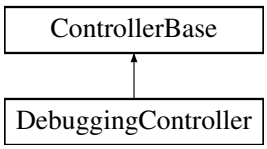
The documentation for this class was generated from the following file:

- DbInitializer.cs

4.9 DebuggingController Class Reference

Debugging controller, only available in development mode. Used for the unit tests.

Inheritance diagram for DebuggingController:



Public Member Functions

- **DebuggingController** (InertiaContext db, IWebHostEnvironment env)
- async Task< ActionResult > **RemoveAccount** (string email)
- async Task< ActionResult > **RemoveOrder** (string orderId)

4.9.1 Detailed Description

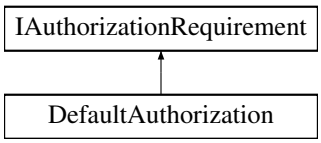
Debugging controller, only available in development mode. Used for the unit tests.

The documentation for this class was generated from the following file:

- Controllers/DebuggingController.cs

4.10 DefaultAuthorization Class Reference

Inheritance diagram for DefaultAuthorization:



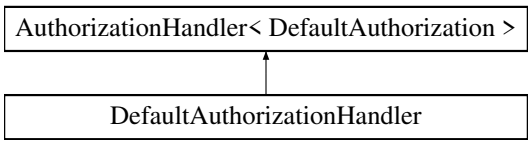
The documentation for this class was generated from the following file:

- Authorization/DefaultAuthorization.cs

4.11 DefaultAuthorizationHandler Class Reference

Handles whether the user is logged in with a valid access token

Inheritance diagram for DefaultAuthorizationHandler:



Public Member Functions

- **DefaultAuthorizationHandler** (InertiaContext db, IHttpContextAccessor httpContextAccessor)

Protected Member Functions

- override async Task **HandleRequirementAsync** (AuthorizationHandlerContext context, DefaultAuthorization requirement)

4.11.1 Detailed Description

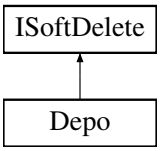
Handles whether the user is logged in with a valid access token

The documentation for this class was generated from the following file:

- Authorization/DefaultAuthorizationHandler.cs

4.12 Depo Class Reference

Inheritance diagram for Depo:



Properties

- int **DepoId** `[get, set]`
- string **Name** = null! `[get, set]`
- float **Latitude** `[get, set]`
- float **Longitude** `[get, set]`
- string **Address** = null! `[get, set]`
- bool **SoftDeleted** `[get, set]`

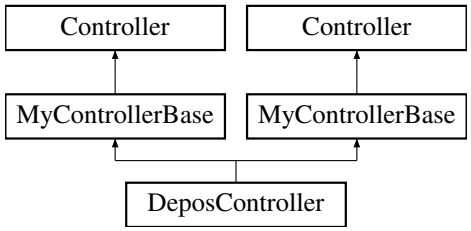
The documentation for this class was generated from the following file:

- Models/Depo.cs

4.13 DeposController Class Reference

Admin controller for modifying, adding and removing depos.

Inheritance diagram for DeposController:



Public Member Functions

- **DeposController** (InertiaContext db)
- async Task< Depo > AddItem ([FromBody] Dtos.CreateDepoRequest depo)

  Endpoint that creates a new Depo
- async Task< ActionResult > RemoveItem (int id)

  Endpoint that removes a depo.
- async Task< ActionResult > UpdateItem (int id, [FromBody] PatchDepoRequest depoRequest)

  Endpoint that modifies a depo
- **DeposController** (InertiaContext db)
- async Task< IEnumerable< Depo > > List ()

  Lists all the depos
- async Task< ActionResult > GetItem (int id)

  Returns all the details of a depo, by id.

Additional Inherited Members

4.13.1 Detailed Description

Admin controller for modifying, adding and removing depos.

Depos controller, to list and search through depos.

4.13.2 Member Function Documentation

4.13.2.1 AddItem()

```
async Task<Depo> DeposController.AddItem (
            [FromBody] Dtos.CreateDepoRequest depo )  [inline]
```

Endpoint that creates a new Depo

Parameters

depo	

Returns

4.13.2.2 GetItem()

```
async Task<ActionResult> DeposController.GetItem (
            int id )  [inline]
```

Returns all the details of a depo, by id.

Parameters

id	

Returns

4.13.2.3 List()

```
async Task<IEnumerable<Depo> > DeposController.List ( )  [inline]
```

Lists all the depos

Returns

4.13.2.4 RemoveItem()

```
async Task<ActionResult> DeposController.RemoveItem (
            int id )  [inline]
```

Endpoint that removes a depo.

Parameters

id	

Returns

4.13.2.5 UpdateItem()

```
async Task<ActionResult> DeposController.UpdateItem (
            int id,
            [FromBody] PatchDepoRequest depoRequest )  [inline]
```

Endpoint that modifies a depo

Parameters

id	
depoRequest	

Returns

The documentation for this class was generated from the following file:

- Controllers/Admin/DeposController.cs

4.14 DiscountApplication Class Reference

Properties

- int **DiscountApplicationId** `[get, set]`
- string **AccountId** = null! `[get, set]`
- virtual Account **Account** = null! `[get, set]`
- UserType **DisccountType** `[get, set]`
- DiscountApplicationState **State** `[get, set]`
- byte?[] **Image** `[get, set]`

The documentation for this class was generated from the following file:

- Models/DiscountApplication.cs

4.15 DiscountApplicationModel Class Reference

Properties

- string **Name** = null! `[get, set]`
- string **DiscountType** = null! `[get, set]`

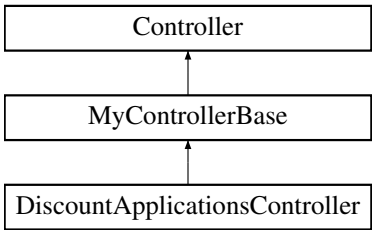
The documentation for this class was generated from the following file:

- Views/DiscountApplicationModel.cs

4.16 DiscountApplicationsController Class Reference

Admin controller that implements approving and denying discount applications

Inheritance diagram for DiscountApplicationsController:



Public Member Functions

- **DiscountApplicationsController** (InertiaContext db, EmailService email)
- async Task< ActionResult > List ()

  Fetches a list of the current discount applications
- async Task< ActionResult > GetImage (int applicationId)

  Fetches the image of a discount application, by id.
- async Task< ActionResult > Approve (int applicationId)

  Approved a discount application, by id.
- async Task< ActionResult > Deny (int applicationId)

  Rejects a discount application by id.

Additional Inherited Members

4.16.1 Detailed Description

Admin controller that implements approving and denying discount applications

4.16.2 Member Function Documentation

4.16.2.1 Approve()

```
async Task<ActionResult> DiscountApplicationsController.Approve (
            int applicationId )  [inline]
```

Approved a discount application, by id.

Parameters

applicationId	

Returns

4.16.2.2 Deny()

```
async Task<ActionResult> DiscountApplicationsController.Deny (
            int applicationId )  [inline]
```

Rejects a discount application by id.

Parameters

applicationId	

Returns

4.16.2.3 GetImage()

```
async Task<ActionResult> DiscountApplicationsController.GetImage (
            int applicationId )  [inline]
```

Fetches the image of a discount application, by id.

Parameters

applicationId	

Returns

4.16.2.4 List()

```
async Task<ActionResult> DiscountApplicationsController.List ( )  [inline]
```

Fetches a list of the current discount applications

Returns

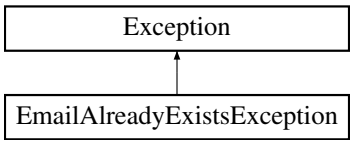
The documentation for this class was generated from the following file:

- Controllers/Admin/DiscountApplicationsController.cs

4.17 EmailAlreadyExistsException Class Reference

Thrown when an account cannot be created due to the email address being already taken.

Inheritance diagram for EmailAlreadyExistsException:



4.17.1 Detailed Description

Thrown when an account cannot be created due to the email address being already taken.

The documentation for this class was generated from the following file:

- Exceptions/Exceptions.cs

4.18 EmailService Class Reference

The service that handles templating the email content and sending the emails

Public Member Functions

- **EmailService** (IRazorViewEngine razorViewEngine, ITempDataProvider tempDataProvider, IService↩ Provider serviceProvider, IConfiguration configuration, InertiaContext db, ILogger< EmailService > logger)
- async Task SendOrderConfirmation (string email, Order order)

  Renders the render confirmation email and sends to the user.
- async Task SendOrderApproval (string email, Order order)

  Renders the render order approval email and sends to the user.
- async Task SendOrderDenied (string email, Order order)

  Renders the order denied email and sends to the user.
- async Task SendOrderCancellation (string email, Order order)

  Renders the order cancellation email and sends to the user.
- async Task SendOrderExtension (string email, Order order)

  Renders the order was extended email and sends to the user.
- async Task SendDiscountApplication (string email, Account account)

  Renders the successful discount application email and sends to the user.
- async Task SendSignup (string email, Account account)

  Renders the signup confirmation email and sends to the user.

4.18.1 Detailed Description

The service that handles templating the email content and sending the emails

4.18.2 Member Function Documentation

4.18.2.1 SendDiscountApplication()

```
async Task EmailService.SendDiscountApplication (
            string email,
            Account account )  [inline]
```

Renders the successful discount application email and sends to the user.

Parameters

email	
account	

4.18.2.2 SendOrderApproval()

```
async Task EmailService.SendOrderApproval (
            string email,
            Order order )  [inline]
```

Renders the render order approval email and sends to the user.

Parameters

email	
order	

4.18.2.3 SendOrderCancellation()

```
async Task EmailService.SendOrderCancellation (
            string email,
            Order order )  [inline]
```

Renders the order cancellation email and sends to the user.

Parameters

email	
order	

4.18.2.4 SendOrderConfirmation()

```
async Task EmailService.SendOrderConfirmation (
            string email,
            Order order )  [inline]
```

Renders the render confirmation email and sends to the user.

Parameters

email	
order	

4.18.2.5 SendOrderDenied()

```
async Task EmailService.SendOrderDenied (
            string email,
            Order order )  [inline]
```

Renders the order denied email and sends to the user.

Parameters

email	
order	

4.18.2.6 SendOrderExtension()

```
async Task EmailService.SendOrderExtension (
            string email,
            Order order )  [inline]
```

Renders the order was extended email and sends to the user.

Parameters

email	
order	

4.18.2.7 SendSignup()

```
async Task EmailService.SendSignup (
            string email,
            Account account )  [inline]
```

Renders the signup confirmation email and sends to the user.

Parameters

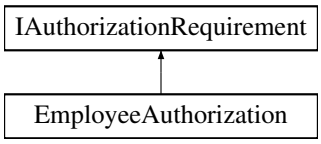
email	
account	

The documentation for this class was generated from the following file:

- Services/EmailService.cs

4.19 EmployeeAuthorization Class Reference

Inheritance diagram for EmployeeAuthorization:



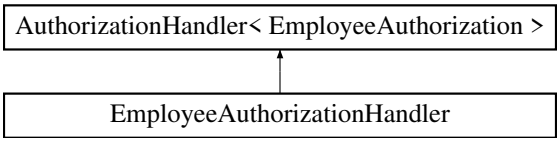
The documentation for this class was generated from the following file:

- Authorization/EmployeeAuthorization.cs

4.20 EmployeeAuthorizationHandler Class Reference

Handles Authorization requirement that checks for employee role.

Inheritance diagram for EmployeeAuthorizationHandler:



Public Member Functions

- **EmployeeAuthorizationHandler** (IHttpContextAccessor httpContextAccessor)

Protected Member Functions

- override Task **HandleRequirementAsync** (AuthorizationHandlerContext context, EmployeeAuthorization requirement)

4.20.1 Detailed Description

Handles Authorization requirement that checks for employee role.

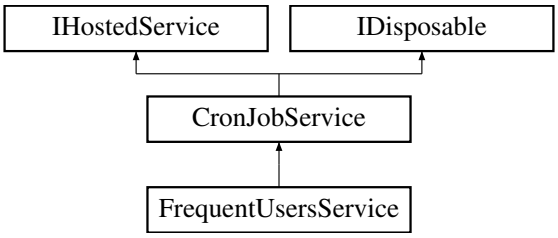
The documentation for this class was generated from the following file:

- Authorization/EmployeeAuthorizationHandler.cs

4.21 FrequentUsersService Class Reference

Recurring task that calculates the Frequent Users every week.

Inheritance diagram for FrequentUsersService:



Public Member Functions

- **FrequentUsersService** (IScheduleConfig< FrequentUsersService > config, ILogger< FrequentUsersService > logger, IServiceScopeFactory factory)
- override Task **StartAsync** (CancellationToken token)
- override async Task **DoWork** (CancellationToken token)
- override Task **StopAsync** (CancellationToken token)

Additional Inherited Members

4.21.1 Detailed Description

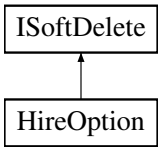
Recurring task that calculates the Frequent Users every week.

The documentation for this class was generated from the following file:

- Services/FrequentUsersService.cs

4.22 HireOption Class Reference

Inheritance diagram for HireOption:



Properties

- int **HireOptionId** `[get, set]`
- int **DurationInHours** `[get, set]`
- string **Name** = null! `[get, set]`
- float **Cost** `[get, set]`
- bool **SoftDeleted** `[get, set]`

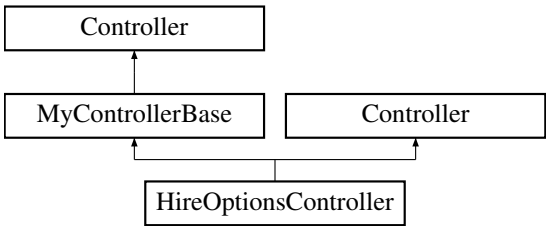
The documentation for this class was generated from the following file:

- Models/HireOption.cs

4.23 HireOptionsController Class Reference

Admin Controller that creates, modifies and removes hire options.

Inheritance diagram for HireOptionsController:



Public Member Functions

- **HireOptionsController** (InertiaContext db)
- async Task< ActionResult > List ()

  lists all hire options
- async Task< ActionResult > Get (int hireOptionId)

  searches for a hire option, by id
- async Task< ActionResult > Create ([FromBody] CreateHireOptionRequest request)

  creates a hire option
- async Task< ActionResult > Edit (int hireOptionId, [FromBody] PatchHireOptionRequest request)

  modifies a hire option, by id
- async Task< ActionResult > Remove (int hireOptionId)

  removes a hire option, by id.
- **HireOptionsController** (InertiaContext db)
- async Task< ActionResult > List ()

  Returns a list of all hire options, available for usage.

Additional Inherited Members

4.23.1 Detailed Description

Admin Controller that creates, modifies and removes hire options.

Controller to check all the currently available hire options

4.23.2 Member Function Documentation

4.23.2.1 Create()

```
async Task<ActionResult> HireOptionsController.Create (
            [FromBody] CreateHireOptionRequest request )  [inline]
```

creates a hire option

Parameters

request	

Returns

4.23.2.2 Edit()

```
async Task<ActionResult> HireOptionsController.Edit (
            int hireOptionId,
            [FromBody] PatchHireOptionRequest request )  [inline]
```

modifies a hire option, by id

Parameters

hire↩ OptionId	
request	

Returns

4.23.2.3 Get()

```
async Task<ActionResult> HireOptionsController.Get (
            int hireOptionId )  [inline]
```

searches for a hire option, by id

Parameters

hire↩ OptionId	

Returns

4.23.2.4 List() [1/2]

```
async Task<ActionResult> HireOptionsController.List ( )  [inline]
```

lists all hire options

Returns

4.23.2.5 List() [2/2]

```
async Task<ActionResult> HireOptionsController.List ( )  [inline]
```

Returns a list of all hire options, available for usage.

Returns

4.23.2.6 Remove()

```
async Task<ActionResult> HireOptionsController.Remove (
            int hireOptionId )  [inline]
```

removes a hire option, by id.

Parameters

hire↩ OptionId	

Returns

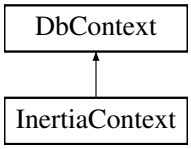
The documentation for this class was generated from the following file:

- Controllers/Admin/HireOptionsController.cs

4.24 InertiaContext Class Reference

Database context of the application. All tables are generated from this class by Entity Framework

Inheritance diagram for InertiaContext:



Public Member Functions

- **InertiaContext** (DbContextOptions options)

Protected Member Functions

- override void **OnConfiguring** (DbContextOptionsBuilder optionsBuilder)
- override void **OnModelCreating** (ModelBuilder modelBuilder)

Properties

- DbSet< Depo > Depos = null! `[get, set]`
  Table for Depos
- DbSet< Scooter > Scooters = null! `[get, set]`
  Table for Scooters
- DbSet< Account > Accounts = null! `[get, set]`
  Table for Accounts
- DbSet< LoginInstance > LoginInstances = null! `[get, set]`
  Table for LoginInstances
- DbSet< Order > Orders = null! `[get, set]`
  Table for Orders
- DbSet< HireOption > HireOptions = null! `[get, set]`
  Table for HireOptions
- DbSet< Issue > Issues = null! `[get, set]`
  Table for Issues
- DbSet< DiscountApplication > DiscountApplications = null! `[get, set]`
  Table for DiscountApplications

4.24.1 Detailed Description

Database context of the application. All tables are generated from this class by Entity Framework

4.24.2 Property Documentation

4.24.2.1 Accounts

```
DbSet<Account> InertiaContext.Accounts = null!  [get], [set]
```

Table for Accounts

4.24.2.2 Depos

```
DbSet<Depo> InertiaContext.Depos = null!  [get], [set]
```

Table for Depos

4.24.2.3 DiscountApplications

```
DbSet<DiscountApplication> InertiaContext.DiscountApplications = null!  [get], [set]
```

Table for DiscountApplications

4.24.2.4 HireOptions

```
DbSet<HireOption> InertiaContext.HireOptions = null!  [get], [set]
```

Table for HireOptions

4.24.2.5 Issues

```
DbSet<Issue> InertiaContext.Issues = null!  [get], [set]
```

Table for Issues

4.24.2.6 LoginInstances

```
DbSet<LoginInstance> InertiaContext.LoginInstances = null!  [get], [set]
```

Table for LoginInstances

4.24.2.7 Orders

```
DbSet<Order> InertiaContext.Orders = null!  [get], [set]
```

Table for Orders

4.24.2.8 Scooters

```
DbSet<Scooter> InertiaContext.Scooters = null!  [get], [set]
```

Table for Scooters

The documentation for this class was generated from the following file:

- InertiaContext.cs

4.25 InertiaService Class Reference

Service that takes care of keeping Scooter states up to date according with the current ongoing bookings.

Public Member Functions

- **InertiaService** (InertiaContext db)
- async Task< IEnumerable< Scooter > > GetAllScootersCurrentStatus (Depo? depo=null)

  Computes the current status of all scooters
- async Task< IEnumerable< Scooter > > GetAvailableScooters (DateTime startTime, DateTime endTime)

  Checks all the scooters that are not booked by another order, from `startTime` to `endTime`
- async Task< IEnumerable< Scooter > > GetAvailableScooters (Depo depo, DateTime startTime, DateTime? endTime=null)

  Checks all the scooters that are not booked by another order, from `startTime` to `endTime`, and that are in the depot `depo`.
- async Task< bool > IsScooterAvailable (Scooter scooter, DateTime startTime, DateTime? endTime=null)

  Checks whether a scooter is available (not booked) between `startTime` and `endTime`.
- async Task< bool > IsScooterAvailableForExtension (Order toExtend, Scooter scooter, DateTime startTime, DateTime? endTime=null)

  Checks whether an order can be extended.
- async Task ReturnScooter (Scooter scooter)

  Marks a scooter as returned, by marking the booking that was using the scooter that it returned the scooter.

- async Task UpdateOrderStatus ()

  Updates all the order status, promoting them from Upcoming, to Ongoing, to PendingReturn depending on the current time.
- async Task< Order > CreateOrder (Account account, Scooter scooter, HireOption hireOption, DateTime startTime)

  Creates an order
- async Task< Order > ExtendOrder (Order order, HireOption hireOption)

  Extends an order
- async Task CancelOrder (Order order)

  Cancels an order
- async Task UpdateFrequentCustomers ()

  Computes a list of users that are frequent and promotes them to frequent users for that week. Used from FrequentUsersService.

4.25.1 Detailed Description

Service that takes care of keeping Scooter states up to date according with the current ongoing bookings.

4.25.2 Member Function Documentation

4.25.2.1 CancelOrder()

```
async Task InertiaService.CancelOrder (
            Order order )  [inline]
```

Cancels an order

Parameters

order	

Exceptions

OrderApprovedOrOngoingException	

4.25.2.2 CreateOrder()

```
async Task<Order> InertiaService.CreateOrder (
            Account account,
            Scooter scooter,
            HireOption hireOption,
            DateTime startTime )  [inline]
```

Creates an order

Parameters

account	
scooter	
hireOption	
startTime	

Returns

Exceptions

UnavailableScooterException	

4.25.2.3 ExtendOrder()

```
async Task<Order> InertiaService.ExtendOrder (
            Order order,
            HireOption hireOption )  [inline]
```

Extends an order

Parameters

order	
hireOption	

Returns

Exceptions

OrderCannotBeExtendException	
UnavailableScooterException	

4.25.2.4 GetAllScootersCurrentStatus()

```
async Task<IEnumerable<Scooter> > InertiaService.GetAllScootersCurrentStatus (
            Depo?  depo = null )  [inline]
```

Computes the current status of all scooters

Parameters

depo	filters the scooters that are in `depo`

Returns

List of scooters with the up to date status

4.25.2.5 GetAvailableScooters() [1/2]

```
async Task<IEnumerable<Scooter> > InertiaService.GetAvailableScooters (
            DateTime startTime,
            DateTime endTime )  [inline]
```

Checks all the scooters that are not booked by another order, from `startTime` to `endTime`

Parameters

startTime	
endTime	

Returns

4.25.2.6 GetAvailableScooters() [2/2]

```
async Task<IEnumerable<Scooter> > InertiaService.GetAvailableScooters (
            Depo depo,
            DateTime startTime,
            DateTime?  endTime = null )  [inline]
```

Checks all the scooters that are not booked by another order, from `startTime` to `endTime`, and that are in the depot `depo`.

Parameters

depo	
startTime	
endTime	

Returns

4.25.2.7 IsScooterAvailable()

```
async Task<bool> InertiaService.IsScooterAvailable (
            Scooter scooter,
            DateTime startTime,
            DateTime?  endTime = null )  [inline]
```

Checks whether a scooter is available (not booked) between `startTime` and `endTime`.

Parameters

scooter	
startTime	
endTime	

Returns

4.25.2.8 IsScooterAvailableForExtension()

```
async Task<bool> InertiaService.IsScooterAvailableForExtension (
            Order toExtend,
            Scooter scooter,
            DateTime startTime,
            DateTime?  endTime = null )  [inline]
```

Checks whether an order can be extended.

Parameters

toExtend	
scooter	
startTime	
endTime	

Returns

4.25.2.9 ReturnScooter()

```
async Task InertiaService.ReturnScooter (
            Scooter scooter )  [inline]
```

Marks a scooter as returned, by marking the booking that was using the scooter that it returned the scooter.

Parameters

scooter	

4.25.2.10 UpdateFrequentCustomers()

```
async Task InertiaService.UpdateFrequentCustomers ( )  [inline]
```

Computes a list of users that are frequent and promotes them to frequent users for that week. Used from FrequentUsersService.

4.25.2.11 UpdateOrderStatus()

```
async Task InertiaService.UpdateOrderStatus ( )  [inline]
```

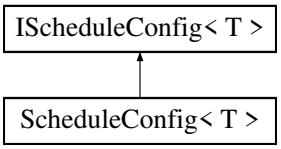
Updates all the order status, promoting them from Upcoming, to Ongoing, to PendingReturn depending on the current time.

The documentation for this class was generated from the following file:

- Services/InertiaService.cs

4.26 IScheduleConfig< T > Interface Template Reference

Inheritance diagram for IScheduleConfig< T >:



Properties

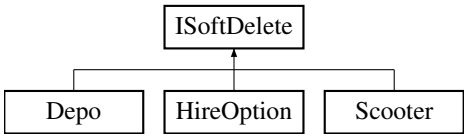
- string **CronExpression** `[get, set]`
- TimeZoneInfo **TimeZoneInfo** `[get, set]`

The documentation for this interface was generated from the following file:

- Services/CronJobService.cs

4.27 ISoftDelete Interface Reference

Inheritance diagram for ISoftDelete:



Properties

- bool **SoftDeleted** `[get, set]`

The documentation for this interface was generated from the following file:

- Models/ISoftDelete.cs

4.28 Issue Class Reference

Properties

- int **IssueId** `[get, set]`
- IssuePriority **Priority** `[get, set]`
- string **Title** = null! `[get, set]`
- string **Content** = null! `[get, set]`
- string **AccountId** = null! `[get, set]`
- virtual Account **Account** = null! `[get, set]`
- DateTime **DateOpened** `[get, set]`
- string? **Resolution** `[get, set]`

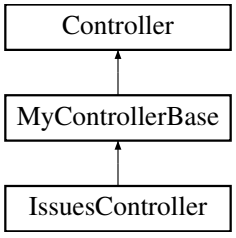
The documentation for this class was generated from the following file:

- Models/Issue.cs

4.29 IssuesController Class Reference

admin controller that implements the issue system.

Inheritance diagram for IssuesController:



Public Member Functions

- **IssuesController** (InertiaContext db)
- async Task< ActionResult< List< Issue > > > ListIssues ([FromQuery] bool? closed, [FromQuery] IssuePriority? priority)

  Lists issues, filtering through them; by default all issues are returned
- async Task< ActionResult > GetIssue (int issueId)

  gets all details of an issue, by id.
- async Task< ActionResult > PatchIssue (int issueId, [FromBody] PatchIssueRequest request)

  Modifies an issue, by id. Used to escalate, deescalate and close issues.

Additional Inherited Members

4.29.1 Detailed Description

admin controller that implements the issue system.

4.29.2 Member Function Documentation

4.29.2.1 GetIssue()

```
async Task<ActionResult> IssuesController.GetIssue (
            int issueId )  [inline]
```

gets all details of an issue, by id.

Parameters

issueId	

Returns

4.29.2.2 ListIssues()

```
async Task<ActionResult<List<Issue> > > IssuesController.ListIssues (
            [FromQuery] bool?  closed,
            [FromQuery] IssuePriority?  priority )  [inline]
```

Lists issues, filtering through them; by default all issues are returned

Parameters

closed	filters by closed issues
priority	filters by priority

Returns

4.29.2.3 PatchIssue()

```
async Task<ActionResult> IssuesController.PatchIssue (
            int issueId,
            [FromBody] PatchIssueRequest request )  [inline]
```

Modifies an issue, by id. Used to escalate, deescalate and close issues.

Parameters

issueId	
request	

Returns

The documentation for this class was generated from the following file:

- Controllers/Admin/IssuesController.cs

4.30 StripeApiController.Item Class Reference

Properties

- string **Id** = null! `[get, set]`

The documentation for this class was generated from the following file:

- Controllers/StripeApiController.cs

4.31 LoginInstance Class Reference

Properties

- int **LoginInstanceId** `[get, set]`
- string **AccessToken** = null! `[get, set]`
- DateTime **CreatedAt** `[get, set]`
- string **AccountId** = null! `[get, set]`
- Account **Account** = null! `[get, set]`
- LoginInstanceState **LoginState** `[get, set]`

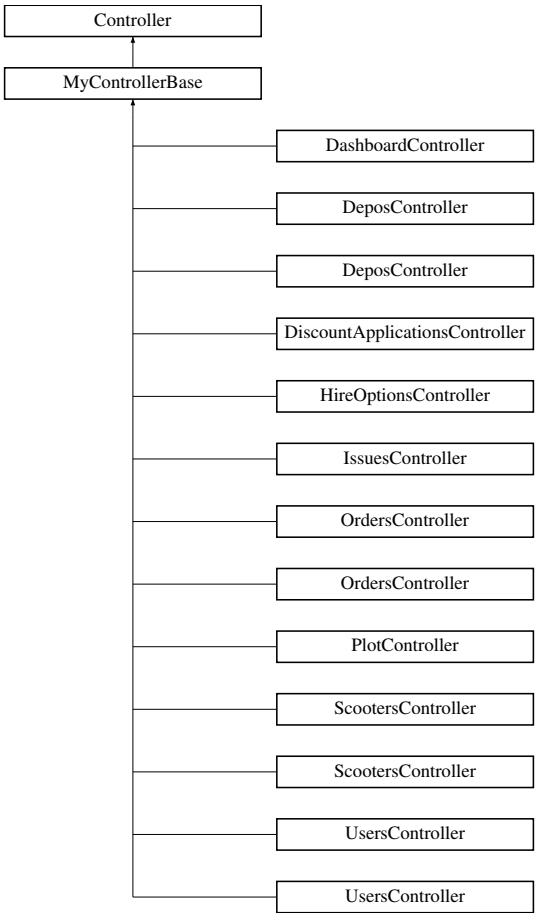
The documentation for this class was generated from the following file:

- Models/LoginInstance.cs

4.32 MyControllerBase Class Reference

Customer Controller base to make returning ApplicationError easier.

Inheritance diagram for MyControllerBase:



Protected Member Functions

- ActionResult **ApplicationError** (ApplicationErrorCode errorCode, string message, string? detail=null)

4.32.1 Detailed Description

Customer Controller base to make returning ApplicationError easier.

The documentation for this class was generated from the following file:

- Controllers/MyControllerBase.cs

4.33 Order Class Reference

Properties

- string **OrderId** = null! `[get, set]`
- int **ScooterId** `[get, set]`
- virtual Scooter **Scooter** = null! `[get, set]`
- DateTime **CreatedAt** `[get, set]`
- DateTime **StartTime** `[get, set]`
- DateTime **EndTime** `[get, set]`
- double **BookTime** `[get, set]`
- long **WeekNumber** `[get, set]`
- DayOfWeek **WeekDay** `[get, set]`
- float **Cost** `[get, set]`
- float **PreDiscountCost** `[get, set]`
- float **Discount** `[get, set]`
- int **HireOptionId** `[get, set]`
- virtual HireOption **HireOption** = null! `[get, set]`
- OrderState **OrderState** `[get, set]`
- string? **ExtendsId** = null! `[get, set]`
- virtual ? Order **Extends** = null `[get, set]`
- ICollection< Order >? **Extensions** = null `[get, set]`
- string **AccountId** = null! `[get, set]`
- virtual Account **Account** = null! `[get, set]`

The documentation for this class was generated from the following file:

- Models/Order.cs

4.34 OrderApprovalModel Class Reference

Properties

- string **Name** = null! `[get, set]`
- string **OrderId** = null! `[get, set]`

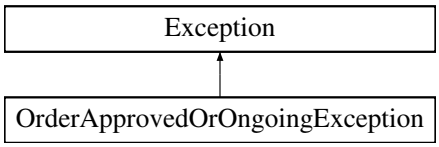
The documentation for this class was generated from the following file:

- Views/OrderApprovalModel.cs

4.35 OrderApprovedOrOngoingException Class Reference

Thrown when an order cannot be cancelled

Inheritance diagram for OrderApprovedOrOngoingException:



4.35.1 Detailed Description

Thrown when an order cannot be cancelled

The documentation for this class was generated from the following file:

- Exceptions/Exceptions.cs

4.36 OrderCancellationModel Class Reference

Properties

- string **Name** = null! `[get, set]`
- int **ScooterId** `[get, set]`
- string **Depo** = null! `[get, set]`
- string **OrderId** = null! `[get, set]`
- float **PreDiscountCost** `[get, set]`
- float **Discount** `[get, set]`
- float **Cost** `[get, set]`
- string **HireOptionName** = null! `[get, set]`

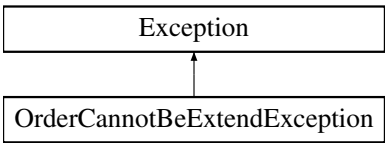
The documentation for this class was generated from the following file:

- Views/OrderCancellationModel.cs

4.37 OrderCannotBeExtendException Class Reference

Thrown when an order cannot be extended

Inheritance diagram for OrderCannotBeExtendException:



4.37.1 Detailed Description

Thrown when an order cannot be extended

The documentation for this class was generated from the following file:

- Exceptions/Exceptions.cs

4.38 OrderConfirmationModel Class Reference

Properties

- string **Name** = null! `[get, set]`
- int **ScooterId** `[get, set]`
- string **Depo** = null! `[get, set]`
- string **OrderId** = null! `[get, set]`
- float **PreDiscountCost** `[get, set]`
- float **Discount** `[get, set]`
- float **Cost** `[get, set]`
- string **HireOptionName** = null! `[get, set]`

The documentation for this class was generated from the following file:

- Views/OrderConfirmationModel.cs

4.39 OrderDeniedModel Class Reference

Properties

- string **Name** = null! `[get, set]`
- string **OrderId** = null! `[get, set]`

The documentation for this class was generated from the following file:

- Views/OrderDeniedModel.cs

4.40 OrderExtensionModel Class Reference

Properties

- string **Name** = null! `[get, set]`
- int **ScooterId** `[get, set]`
- string **Depo** = null! `[get, set]`
- string **OrderId** = null! `[get, set]`
- float **PreDiscountCost** `[get, set]`
- float **Discount** `[get, set]`
- float **Cost** `[get, set]`
- List< Tuple< string, DateTime > > **Extensions** = null! `[get, set]`

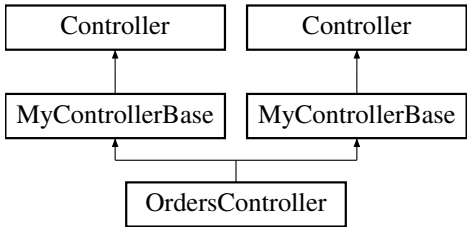
The documentation for this class was generated from the following file:

- Views/OrderExtensionModel.cs

4.41 OrdersController Class Reference

Admin controller for operating on orders.

Inheritance diagram for OrdersController:



Public Member Functions

- **OrdersController** (InertiaContext db, InertiaService inertia, EmailService email, UsersService users)
- async Task< ActionResult< List< Order > > > ListOrders ()

  Returns a list of all the orders, by all users.
- async Task< ActionResult > ListOrders (string orderId)

  gets all the details of an order, by id.
- async Task< ActionResult > CreateGuestOrder ([FromBody] CreateGuestOrderRequest createOrder)

  Creates a guest order, used for employees creating guest orders
- async Task< ActionResult > CancelOrder (string orderId)

  Cancels an order
- async Task< ActionResult > ExtendOrder (string orderId, [FromBody] ExtendOrderRequest extendOrder)

  Extends an order
- async Task< ActionResult > ApproveOrder (string orderId)

  Approves an order
- async Task< ActionResult > DenyOrder (string orderId)

  Rejects an order
- **OrdersController** (InertiaContext db, InertiaService inertia, EmailService email)
- async Task< ActionResult > GetOrder (string orderId)

  Returns all the details of an order, by id.
- async Task< ActionResult > CreateOrder ([FromBody] CreateOrderRequest createOrder)

  endpoint for creating an order.
- async Task< ActionResult > CancelOrder (string orderId)

  endpoint for cancelling an order.
- async Task< ActionResult > ExtendOrder (string orderId, [FromBody] ExtendOrderRequest extendOrder)

  endpoint for extending an order

Additional Inherited Members

4.41.1 Detailed Description

Admin controller for operating on orders.

Controller for Creating, Querying, Extending and Cancelling orders.

4.41.2 Member Function Documentation

4.41.2.1 ApproveOrder()

```
async Task<ActionResult> OrdersController.ApproveOrder (
            string orderId )  [inline]
```

Approves an order

Parameters

orderId	

Returns

4.41.2.2 CancelOrder() [1/2]

```
async Task<ActionResult> OrdersController.CancelOrder (
            string orderId )  [inline]
```

Cancels an order

Parameters

orderId	

Returns

4.41.2.3 CancelOrder() [2/2]

```
async Task<ActionResult> OrdersController.CancelOrder (
            string orderId )  [inline]
```

endpoint for cancelling an order.

Parameters

orderId	

Returns

4.41.2.4 CreateGuestOrder()

```
async Task<ActionResult> OrdersController.CreateGuestOrder (
            [FromBody] CreateGuestOrderRequest createOrder )  [inline]
```

Creates a guest order, used for employees creating guest orders

Parameters

createOrder	

Returns

4.41.2.5 CreateOrder()

```
async Task<ActionResult> OrdersController.CreateOrder (
            [FromBody] CreateOrderRequest createOrder )  [inline]
```

endpoint for creating an order.

Parameters

createOrder	

Returns

4.41.2.6 DenyOrder()

```
async Task<ActionResult> OrdersController.DenyOrder (
            string orderId )  [inline]
```

Rejects an order

Parameters

order↩ Id	

Returns

4.41.2.7 ExtendOrder() [1/2]

```
async Task<ActionResult> OrdersController.ExtendOrder (
            string orderId,
            [FromBody] ExtendOrderRequest extendOrder )  [inline]
```

Extends an order

Parameters

orderId	
extendOrder	

Returns

4.41.2.8 ExtendOrder() [2/2]

```
async Task<ActionResult> OrdersController.ExtendOrder (
            string orderId,
            [FromBody] ExtendOrderRequest extendOrder )  [inline]
```

endpoint for extending an order

Parameters

orderId	
extendOrder	

Returns

4.41.2.9 GetOrder()

```
async Task<ActionResult> OrdersController.GetOrder (
            string orderId )  [inline]
```

Returns all the details of an order, by id.

Parameters

orderId	

Returns

4.41.2.10 ListOrders() [1/2]

```
async Task<ActionResult<List<Order> > > OrdersController.ListOrders ( )  [inline]
```

Returns a list of all the orders, by all users.

Returns

4.41.2.11 ListOrders() [2/2]

```
async Task<ActionResult> OrdersController.ListOrders (
            string orderId )  [inline]
```

gets all the details of an order, by id.

Parameters

orderId	

Returns

The documentation for this class was generated from the following file:

- Controllers/Admin/OrdersController.cs

4.42 StripeApiController.PaymentIntentCreateRequest Class Reference

Properties

- Item[ ] **Items** = null! `[get, set]`

The documentation for this class was generated from the following file:

- Controllers/StripeApiController.cs

4.43 Plot Class Reference

Properties

- List< int > **XAxis** = null! `[get, set]`
- List< PlotLine > **YAxis** = null! `[get, set]`

The documentation for this class was generated from the following file:

- Models/Plot.cs

4.44 PlotBarChart Class Reference

Properties

- List< String > **Tags** = null! `[get, set]`
- List< String > **BarNames** = null! `[get, set]`
- List< List< float > > **BarData** = null! `[get, set]`

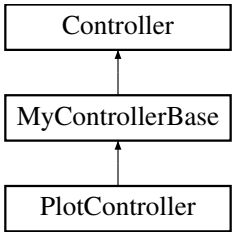
The documentation for this class was generated from the following file:

- Models/PlotBarChart.cs

4.45 PlotController Class Reference

Endpoint for computing the plot values.

Inheritance diagram for PlotController:



Public Member Functions

- **PlotController** (InertiaContext db)
- async Task< ActionResult > PlotWeeklyIncome ([FromQuery] bool separateHireOptions)

  plots weekly income, can separate weekly income by hire options.
- async Task< ActionResult > PlotCombinedDaily ()

  Computes the combined daily plot

Additional Inherited Members

4.45.1 Detailed Description

Endpoint for computing the plot values.

4.45.2 Member Function Documentation

4.45.2.1 PlotCombinedDaily()

```
async Task<ActionResult> PlotController.PlotCombinedDaily ( )  [inline]
```

Computes the combined daily plot

Returns

4.45.2.2 PlotWeeklyIncome()

```
async Task<ActionResult> PlotController.PlotWeeklyIncome (
            [FromQuery] bool separateHireOptions )  [inline]
```

plots weekly income, can separate weekly income by hire options.

Parameters

separateHireOptions	

Returns

The documentation for this class was generated from the following file:

- Controllers/Admin/PlotController.cs

4.46 PlotLine Class Reference

Properties

- string **Name** = null! `[get, set]`
- List< float > **Values** = null! `[get, set]`

The documentation for this class was generated from the following file:

- Models/Plot.cs

4.47 Policies Class Reference

List of policies that can be used an the Authorize attribute

Static Public Attributes

- const string **Authenticated** = "Authenticated"
- const string **MatchAccountId** = "MatchAccountId"
- const string **Employee** = "Employee"

4.47.1 Detailed Description

List of policies that can be used an the Authorize attribute

The documentation for this class was generated from the following file:

- Authorization/Policies.cs

4.48 Roles Class Reference

List of roles

Static Public Attributes

- const string **User** = nameof(AccountRole.User)
- const string **Employee** = nameof(AccountRole.Employee)

4.48.1 Detailed Description

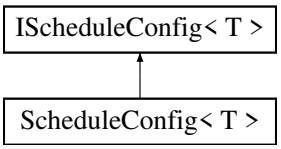
List of roles

The documentation for this class was generated from the following file:

- Authorization/Roles.cs

4.49 ScheduleConfig< T > Class Template Reference

Inheritance diagram for ScheduleConfig< T >:



Properties

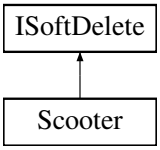
- string **CronExpression** = null! `[get, set]`
- TimeZoneInfo **TimeZoneInfo** = null! `[get, set]`

The documentation for this class was generated from the following file:

- Services/CronJobService.cs

4.50 Scooter Class Reference

Inheritance diagram for Scooter:



Properties

- int **ScooterId** `[get, set]`
- int **SoftScooterId** `[get, set]`
- string **Name** = null! `[get, set]`
- int **DepoId** `[get, set]`
- virtual Depo **Depo** = null! `[get, set]`
- bool **Available** `[get, set]`
- ScooterStatus **ScooterStatus** `[get, set]`
- bool **SoftDeleted** `[get, set]`

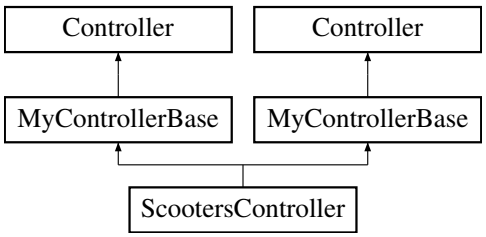
The documentation for this class was generated from the following file:

- Models/Scooter.cs

4.51 ScootersController Class Reference

Admin Controller for handling scooters.

Inheritance diagram for ScootersController:



Public Member Functions

- **ScootersController** (InertiaContext db, InertiaService inertia)
- async Task< ActionResult > List ([FromQuery] int? depoId, [FromQuery] ScooterStatus? status)

  Get a list of all scooters, with their updated status.
- async Task< ActionResult > ReturnScooter (int scooterId)

  Returns a scooter that is in the PendingReturn state.
- async Task< ActionResult > AddItem ([FromBody] CreateScooterRequest request)

  Creates a new scooter
- async Task< ActionResult > RemoveItem (int scooterId)

  removes a scooter
- async Task< ActionResult< Scooter > > UpdateItem (int scooterId, [FromBody] PatchScooterRequest scooterRequest)

  updates the details of a scooter
- **ScootersController** (InertiaContext db, InertiaService service)
- async Task< ActionResult > GetAvailableScooters ([FromQuery(Name="depoId")] int? depoId, [From↩ Query(Name="startTime")] DateTime? startTime, [FromQuery(Name="endTime")] DateTime? endTime)

  Computes and returns a list of the available scooters. Can filter by depoId, and time frame.
- async Task< ActionResult > CountAvailableScooters ([FromQuery(Name="depoId")] int? depoId, [From↩ Query(Name="startTime")] DateTime? startTime, [FromQuery(Name="endTime")] DateTime? endTime)

  Counts the currently available scooters.
- async Task< ActionResult< Scooter > > GetItem (int id)

  Returns the details of a scooter, by id.

Additional Inherited Members

4.51.1 Detailed Description

Admin Controller for handling scooters.

Controller for querying the availability of scooters.

4.51.2 Member Function Documentation

4.51.2.1 AddItem()

```
async Task<ActionResult> ScootersController.AddItem (
            [FromBody] CreateScooterRequest request )  [inline]
```

Creates a new scooter

Parameters

request	

Returns

4.51.2.2 CountAvailableScooters()

```
async Task<ActionResult> ScootersController.CountAvailableScooters (
            [FromQuery(Name = "depoId")] int?  depoId,
            [FromQuery(Name = "startTime")] DateTime?  startTime,
            [FromQuery(Name = "endTime")] DateTime?  endTime )  [inline]
```

Counts the currently available scooters.

Parameters

depoId	
startTime	
endTime	

Returns

4.51.2.3 GetAvailableScooters()

```
async Task<ActionResult> ScootersController.GetAvailableScooters (
            [FromQuery(Name = "depoId")] int?  depoId,
            [FromQuery(Name = "startTime")] DateTime?  startTime,
            [FromQuery(Name = "endTime")] DateTime?  endTime )  [inline]
```

Computes and returns a list of the available scooters. Can filter by depoId, and time frame.

Parameters

depoId	if left empty, queries all depos
startTime	if left empty, queries for the current time
endTime	if left empty, it is set to startTime

Returns

4.51.2.4 GetItem()

```
async Task<ActionResult<Scooter> > ScootersController.GetItem (
            int id )  [inline]
```

Returns the details of a scooter, by id.

Parameters

id	

Returns

4.51.2.5 List()

```
async Task<ActionResult> ScootersController.List (
            [FromQuery] int?  depoId,
            [FromQuery] ScooterStatus?  status )  [inline]
```

Get a list of all scooters, with their updated status.

Parameters

depo↩ Id	
status	

Returns

4.51.2.6 RemoveItem()

```
async Task<ActionResult> ScootersController.RemoveItem (
            int scooterId )  [inline]
```

removes a scooter

Parameters

scooter↩ Id	

Returns

4.51.2.7 ReturnScooter()

```
async Task<ActionResult> ScootersController.ReturnScooter (
            int scooterId )  [inline]
```

Returns a scooter that is in the PendingReturn state.

Parameters

scooter↩ Id	

Returns

4.51.2.8 UpdateItem()

```
async Task<ActionResult<Scooter> > ScootersController.UpdateItem (
            int scooterId,
            [FromBody] PatchScooterRequest scooterRequest )  [inline]
```

updates the details of a scooter

Parameters

scooterId	
scooterRequest	

Returns

The documentation for this class was generated from the following file:

- Controllers/Admin/ScootersController.cs

4.52 SignupModel Class Reference

Properties

- string **Name** = null! `[get, set]`
- string **AccountId** = null! `[get, set]`

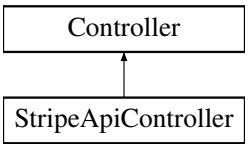
The documentation for this class was generated from the following file:

- Views/SignupModel.cs

4.53 StripeApiController Class Reference

Boiler plate code for integrating with Stripe.

Inheritance diagram for StripeApiController:



Classes

- class Item
- class PaymentIntentCreateRequest

Public Member Functions

- **StripeApiController** (InertiaContext db)
- async Task< ActionResult > **Create** (PaymentIntentCreateRequest request)

4.53.1 Detailed Description

Boiler plate code for integrating with Stripe.

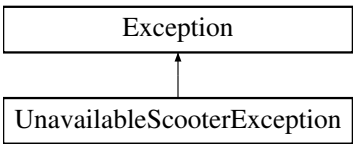
The documentation for this class was generated from the following file:

- Controllers/StripeApiController.cs

4.54 UnavailableScooterException Class Reference

Thrown when a scooter is not available

Inheritance diagram for UnavailableScooterException:



4.54.1 Detailed Description

Thrown when a scooter is not available

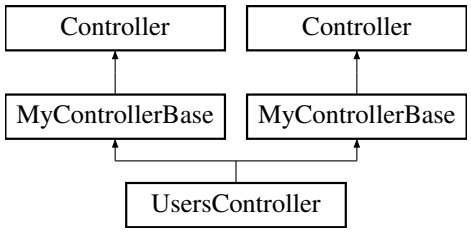
The documentation for this class was generated from the following file:

- Exceptions/Exceptions.cs

4.55 UsersController Class Reference

Admin Controller for operating on users

Inheritance diagram for UsersController:



Public Member Functions

- **UsersController** (InertiaContext db, UsersService users)
- async Task< ActionResult > GetUsers ()

  Gets a list of users.
- async Task< ActionResult > GetUser (string accountId)

  get all the details of a user, by id.
- async Task< ActionResult< CreateUserResponse > > CreateUser ([FromBody] CreateUserRequest request)

  Creates a new user
- async Task< ActionResult > PatchUser (string accountId, [FromBody] PatchUserRequest request)

  Modifies the details of an user account
- **UsersController** (InertiaContext db, AuthenticationTokenService tokenService, InertiaService inertia, UsersService users, EmailService email)

- async Task< ActionResult > Signup ([FromBody] SignupRequest request)

  Allows user to sign up.
- async Task< ActionResult< LoginResponse > > Login ([FromBody] LoginRequest loginRequest)

  Allows user to log in in their account.
- async Task< ActionResult > Logout ([FromBody] LogoutRequest request)

  Allows users to logout (and as such invalidate the access token) from their account.
- async Task< ActionResult > GetProfile (string accountId)

  Returns all the details of the logged in user account.
- async Task< ActionResult > GetOrders (string accountId)

  Returns a list of orders made by the currently logged in user account.
- async Task< ActionResult > GetIssues (string accountId)

  Returns a list of issues made by the currently logged in user account.
- async Task< ActionResult > CreateIssue (string accountId, [FromBody] CreateIssueRequest request)

  Create an issues
- async Task< ActionResult > GetIssue (string accountId, int issueId)

  Get all the details of an issue submitted by the currently logged in user account
- async Task< ActionResult > CloseIssue (string accountId, int issueId)

  Remove an issue submitted by the currently logged in user account
- async Task< ActionResult > ApplyDiscount (string accountId, [FromBody]ApplyDiscountRequest request)

  Apply for discount
- async Task< ActionResult > ApplyDiscountUploadImage (string accountId, [FromBody] byte[ ] image)

  Helper endpoints that allows for uploading images
- async Task< ActionResult > ChangePassword (string accountId, [FromBody] ChangePasswordRequest request)

  Change password for the currently logged in user

Additional Inherited Members

4.55.1 Detailed Description

Admin Controller for operating on users

Controller for providing all account related functionality (applying for discounts, login, signup, modify account)

4.55.2 Member Function Documentation

4.55.2.1 ApplyDiscount()

```
async Task<ActionResult> UsersController.ApplyDiscount (
            string accountId,
            [FromBody] ApplyDiscountRequest request )  [inline]
```

Apply for discount

Parameters

accountId	
request	

Returns

4.55.2.2 ApplyDiscountUploadImage()

```
async Task<ActionResult> UsersController.ApplyDiscountUploadImage (
            string accountId,
            [FromBody] byte[] image )  [inline]
```

Helper endpoints that allows for uploading images

Parameters

accountId	
image	

Returns

4.55.2.3 ChangePassword()

```
async Task<ActionResult> UsersController.ChangePassword (
            string accountId,
            [FromBody] ChangePasswordRequest request )  [inline]
```

Change password for the currently logged in user

Parameters

accountId	
request	

Returns

4.55.2.4 CloseIssue()

```
async Task<ActionResult> UsersController.CloseIssue (
            string accountId,
            int issueId )  [inline]
```

Remove an issue submitted by the currently logged in user account

Parameters

accountId	
issueId	

Returns

4.55.2.5 CreateIssue()

```
async Task<ActionResult> UsersController.CreateIssue (
            string accountId,
            [FromBody] CreateIssueRequest request )  [inline]
```

Create an issues

Parameters

accountId	
request	

Returns

4.55.2.6 CreateUser()

```
async Task<ActionResult<CreateUserResponse> > UsersController.CreateUser (
            [FromBody] CreateUserRequest request )  [inline]
```

Creates a new user

Parameters

request	

Returns

4.55.2.7 GetIssue()

```
async Task<ActionResult> UsersController.GetIssue (
            string accountId,
            int issueId )  [inline]
```

Get all the details of an issue submitted by the currently logged in user account

Parameters

accountId	
issueId	

Returns

4.55.2.8 GetIssues()

```
async Task<ActionResult> UsersController.GetIssues (
            string accountId )  [inline]
```

Returns a list of issues made by the currently logged in user account.

Parameters

accountId	

Returns

4.55.2.9 GetOrders()

```
async Task<ActionResult> UsersController.GetOrders (
            string accountId )  [inline]
```

Returns a list of orders made by the currently logged in user account.

Parameters

accountId	

Returns

4.55.2.10 GetProfile()

```
async Task<ActionResult> UsersController.GetProfile (
            string accountId )  [inline]
```

Returns all the details of the logged in user account.

Parameters

accountId	

Returns

4.55.2.11 GetUser()

```
async Task<ActionResult> UsersController.GetUser (
            string accountId )  [inline]
```

get all the details of a user, by id.

Parameters

accountId	

Returns

4.55.2.12 GetUsers()

```
async Task<ActionResult> UsersController.GetUsers ( )  [inline]
```

Gets a list of users.

Returns

4.55.2.13 Login()

```
async Task<ActionResult<LoginResponse> > UsersController.Login (
            [FromBody] LoginRequest loginRequest )  [inline]
```

Allows user to log in in their account.

Parameters

loginRequest	

Returns

4.55.2.14 Logout()

```
async Task<ActionResult> UsersController.Logout (
            [FromBody] LogoutRequest request )  [inline]
```

Allows users to logout (and as such invalidate the access token) from their account.

Parameters

request	

Returns

4.55.2.15 PatchUser()

```
async Task<ActionResult> UsersController.PatchUser (
            string accountId,
            [FromBody] PatchUserRequest request )  [inline]
```

Modifies the details of an user account

Parameters

accountId	
request	

Returns

4.55.2.16 Signup()

```
async Task<ActionResult> UsersController.Signup (
            [FromBody] SignupRequest request )  [inline]
```

Allows user to sign up.

Parameters

request	

Returns

The documentation for this class was generated from the following file:

- Controllers/Admin/UsersController.cs

4.56 UsersService Class Reference

Service for creating user accounts and logging in users

Public Member Functions

- **UsersService** (InertiaContext db)
- async Task< Account > CreateAccount (string email, string password, string name, UserType userType, AccountRole role)

  Creates user account.
- async Task< Account > ModifyAccount (Account account, string? name, string? email, string? password, AccountRole? accountRole)

  Changes details of a user account, updating all the appropriate data in the database.
- async Task< Account?> MatchAccount (string email, string password)

  Checks whether the email password login information is correct.

Static Public Member Functions

- static string **GeneratePassword** ()

4.56.1 Detailed Description

Service for creating user accounts and logging in users

4.56.2 Member Function Documentation

4.56.2.1 CreateAccount()

```
async Task<Account> UsersService.CreateAccount (
            string email,
            string password,
            string name,
            UserType userType,
            AccountRole role )  [inline]
```

Creates user account.

Parameters

email	
password	
name	
userType	
role	

Returns

Exceptions

EmailAlreadyExistsException	

4.56.2.2 MatchAccount()

```
async Task<Account?> UsersService.MatchAccount (
            string email,
            string password )  [inline]
```

Checks whether the email password login information is correct.

Parameters

email	
password	

Returns

4.56.2.3 ModifyAccount()

```
async Task<Account> UsersService.ModifyAccount (
            Account account,
            string?  name,
            string?  email,
            string?  password,
            AccountRole?  accountRole )  [inline]
```

Changes details of a user account, updating all the appropriate data in the database.

Parameters

account	
name	
email	
password	
accountRole	

Returns

Exceptions

EmailAlreadyExistsException	

The documentation for this class was generated from the following file:

- Services/UsersService.cs

Index