**Naked Functions**

**Application Developer Manual**

**[under construction]**

Contents

[What’s new in Version 12 10](#_Toc77843137)

[Getting started with the Naked Objects Template solution 11](#_Toc77843138)

[The Template Server solution 11](#_Toc77843139)

[The Template Client solution 12](#_Toc77843140)

[Troubleshooting 13](#_Toc77843141)

[Updating the packages 14](#_Toc77843142)

[Exploring the Template domain code 15](#_Toc77843143)

[Writing a new application 16](#_Toc77843144)

[The Naked Objects Client 17](#_Toc77843145)

[Using the generic client 17](#_Toc77843146)

[Configuring the generic client 17](#_Toc77843147)

[Configuring colours for objects 18](#_Toc77843148)

[Configuring masks 20](#_Toc77843149)

[Shortening Urls 21](#_Toc77843150)

[Obfuscating or encrypting sensitive data in client URL query strings 22](#_Toc77843151)

[Other configurable client elements 23](#_Toc77843152)

[Customising the generic client using Angular 24](#_Toc77843153)

[Adding custom views 25](#_Toc77843154)

[The Naked Objects Server 28](#_Toc77843155)

[Application configuration 28](#_Toc77843156)

[Configuring concurrency-checking 29](#_Toc77843157)

[Configuring the EntityObjectStore 29](#_Toc77843158)

[Configuring Authorization 29](#_Toc77843159)

[Configuring Auditing 30](#_Toc77843160)

[Configuring Profiling 31](#_Toc77843161)

[Configuring the user interface 31](#_Toc77843162)

[Configuring the Restful API 31](#_Toc77843163)

[Configuring the RestRoot 32](#_Toc77843164)

[How to determine whether an action will require a GET, PUT or POST method 32](#_Toc77843165)

[How to implement automatic redirection for specific objects 32](#_Toc77843166)

[Cache settings 33](#_Toc77843167)

[How to change the format of the Object Identifier (Oid) in resource URLs 34](#_Toc77843168)

[How to limit the scope of the domain model that is visible through the Restful API 35](#_Toc77843169)

[Configuring cross-origin resource sharing (CORS) 36](#_Toc77843170)

[How to hook additional functionality into controller methods 37](#_Toc77843171)

[How to inject services or the Domain Object Container into the controller: 37](#_Toc77843172)

[The Naked Objects Programming Model 38](#_Toc77843173)

[Principal Concepts 38](#_Toc77843174)

[Domain object 38](#_Toc77843175)

[Property 38](#_Toc77843176)

[Action 39](#_Toc77843177)

[Menus 40](#_Toc77843178)

[Recognised method 43](#_Toc77843179)

[Recognised attribute 43](#_Toc77843180)

[View Model 43](#_Toc77843181)

[Service 46](#_Toc77843182)

[Factories and Repositories 46](#_Toc77843183)

[External or System service 47](#_Toc77843184)

[Contributed action 48](#_Toc77843185)

[Injection of domain services into domain objects 50](#_Toc77843186)

[The Domain Object Container 51](#_Toc77843187)

[Recognised types, attributes, and conventions 53](#_Toc77843188)

[Recognised Value Types 53](#_Toc77843189)

[Recognised Collection types 54](#_Toc77843190)

[Recognised .NET attributes 55](#_Toc77843191)

[Recognised Methods 58](#_Toc77843192)

[NakedObjects.Attributes 61](#_Toc77843193)

[NakedObjects.Types 71](#_Toc77843194)

[NakedObjects.Helpers 72](#_Toc77843195)

[Using the Naked Objects Snippets 74](#_Toc77843196)

[A how-to guide 77](#_Toc77843197)

[The object life-cycle 77](#_Toc77843198)

[How to create an object 77](#_Toc77843199)

[How to persist an object 78](#_Toc77843200)

[How to update an object 78](#_Toc77843201)

[How to delete an object 78](#_Toc77843202)

[How to retrieve existing instances 79](#_Toc77843203)

[How to insert behaviour into the object life cycle 79](#_Toc77843204)

[How to specify that an object should never be persisted 79](#_Toc77843205)

[How to specify that an object should not be modified by the user 79](#_Toc77843206)

[How to specify that a class of objects has a limited number of instances 80](#_Toc77843207)

[How to handle concurrency checking 80](#_Toc77843208)

[Object presentation 81](#_Toc77843209)

[How to specify a title for an object 81](#_Toc77843210)

[How to control the layout of an object’s action menu 82](#_Toc77843211)

[How to specify a name and/or description for an object 82](#_Toc77843212)

[How to specify that an object should be always hidden from the user 82](#_Toc77843213)

[Properties 82](#_Toc77843214)

[How to add a property to a domain object 82](#_Toc77843215)

[How to prevent the user from modifying a property 83](#_Toc77843216)

[How to make a property optional (when saving an object) 84](#_Toc77843217)

[How to specify the size of String properties 84](#_Toc77843218)

[How to validate user input to a property 84](#_Toc77843219)

[How to validate user input to more than one property 85](#_Toc77843220)

[How to specify a default value for a property 86](#_Toc77843221)

[How to specify a set of choices for a property 87](#_Toc77843222)

[How to specify auto-complete for a property 88](#_Toc77843223)

[How to set up the initial value of a property programmatically 89](#_Toc77843224)

[How to trigger other behaviour when a property is changed 89](#_Toc77843225)

[How to control the order in which properties are displayed 90](#_Toc77843226)

[How to specify a name and/or description for a property 90](#_Toc77843227)

[How to hide a property from the user 90](#_Toc77843228)

[How to make a property non-persisted 91](#_Toc77843229)

[How to handle File Attachments 92](#_Toc77843230)

[How to display an image 93](#_Toc77843231)

[How to handle enum properties 93](#_Toc77843232)

[How to work with date properties 94](#_Toc77843233)

[Collection properties 94](#_Toc77843234)

[How to add a collection property to a domain object 95](#_Toc77843235)

[Adding-to or removing objects from a collection 95](#_Toc77843236)

[How to create a derived collection 96](#_Toc77843237)

[How to control the order in which table rows are displayed 97](#_Toc77843238)

[How to specify which columns are displayed in a table view 98](#_Toc77843239)

[How to create an action that operates on a selection from a collection 98](#_Toc77843240)

[Actions 99](#_Toc77843241)

[How to add an action to an object 99](#_Toc77843242)

[How to specify the layout of the menu of actions on an object 99](#_Toc77843243)

[How to define a contributed action 99](#_Toc77843244)

[How to prevent a service action from being a contributed to objects 99](#_Toc77843245)

[How to specify parameter names and/or descriptions 100](#_Toc77843246)

[How to make a parameter optional 100](#_Toc77843247)

[How to permanently disable a parameter optional 100](#_Toc77843248)

[How to specify a default value for a parameter 100](#_Toc77843249)

[How to specify a set of choices for a parameter 101](#_Toc77843250)

[How to allow selection of multiple choices 103](#_Toc77843251)

[How to specify auto-complete for a parameter 103](#_Toc77843252)

[How to specify the length or format for text-input parameters 104](#_Toc77843253)

[How to obscure input text (e.g. for a Password) 104](#_Toc77843254)

[How to validate parameter values 104](#_Toc77843255)

[How to work with date parameters 105](#_Toc77843256)

[How to specify conditions for invoking an action 106](#_Toc77843257)

[How to control the order in which actions appear on the menu 107](#_Toc77843258)

[How to hide actions 107](#_Toc77843259)

[How to pass a message back to the user 108](#_Toc77843260)

[How to work with transactions 108](#_Toc77843261)

[How to create an action that operates on an object’s collection 108](#_Toc77843262)

[How to make an action appear within an object collection 109](#_Toc77843263)

[How to create an action that will allow multiple rows of data to be entered 109](#_Toc77843264)

[Authentication 110](#_Toc77843265)

[Authorization 111](#_Toc77843266)

[Attribute-based Authorization 112](#_Toc77843267)

[Custom Authorization 113](#_Toc77843268)

[Adding further security measures 114](#_Toc77843269)

[Auditing 115](#_Toc77843270)

[Profiling 116](#_Toc77843271)

[Other how-to’s 117](#_Toc77843272)

[How to get hold of the current user programmatically 117](#_Toc77843273)

[How to create an XML Snapshot of an object 118](#_Toc77843274)

[How to run Naked Objects as a .exe 119](#_Toc77843275)

[How to run multiple threads asynchronously 119](#_Toc77843276)

[Patterns and practices 121](#_Toc77843277)

[Working with Entity Framework 121](#_Toc77843278)

[Overriding the default database schema generation 121](#_Toc77843279)

[Using data fixtures with Code First 122](#_Toc77843280)

[How to specify 'eager loading' of an object's reference properties 123](#_Toc77843281)

[How to implement complex types 123](#_Toc77843282)

[How to work with multiple databases 124](#_Toc77843283)

[How to work with multiple database contexts 124](#_Toc77843284)

[Writing safe LINQ queries 126](#_Toc77843285)

[Polymorphic Associations 128](#_Toc77843286)

[Result interface association 128](#_Toc77843287)

[Polymorphic Association 129](#_Toc77843288)

[The cluster pattern 135](#_Toc77843289)

[Building the framework from source 139](#_Toc77843290)

[Troubleshooting 140](#_Toc77843291)

[Logging 140](#_Toc77843292)

[Logging from within the domain code 140](#_Toc77843293)

[Errors thrown when starting an application 141](#_Toc77843294)

[No known services 141](#_Toc77843295)

[Unable to infer a key 141](#_Toc77843296)

[Class not public 141](#_Toc77843297)

[Errors thrown when running an application 141](#_Toc77843298)

[A property is not virtual/overrideable 142](#_Toc77843299)

[Invalid column name 142](#_Toc77843300)

[Invalid object name 142](#_Toc77843301)

[Collection not initialised 142](#_Toc77843302)

[Database is generated, but certain (or all) tables are not being generated 142](#_Toc77843303)

[Unexpected behaviour in the user interface 143](#_Toc77843304)

[A public method is not appearing as an object action 143](#_Toc77843305)

[Default, Choices, Validate or other complementary methods are showing up as menu actions 143](#_Toc77843306)

[Debugging 144](#_Toc77843307)

[Life Cycle methods are not being called 144](#_Toc77843308)

[The injected Container is null 144](#_Toc77843309)

[An injected service is null 144](#_Toc77843310)

# Introduction

## What is Naked Functions

A framework for creating enterprise scale applications built on top of relational database and running on the .NET platform.

Where your domain code follows a pure functional model (i.e. side effect free functions and immutable types) with the benefits that confers [list]

This is possible because it is not necessary to write any I/O code at all.

* Persistence to the database is handled by Microsoft’s EF Core, following the ‘code first’ pattern.
* The NF server generates a complete and truly RESTful API providing full access to all the data types and all the public functions – or, if Authentication and Authorization services are added, offering access only to those types, instances, and functions that the user is authorized to see/use.
* Similarly, an OOUI (meaning…) is generated automatically from your domain model.

Relationship to Naked Objects.

## Starting from the Naked Functions Template solution

Explain what it is and then how to run it and use it

Explore it.

### Server solution

#### Domain model program(s)

Explain default elements incl. the Model Config and the DbContext

#### Server solution

Sets up the system using standard Microsoft patterns

Specifies where to find what is needed from the domain model programs.

Link to system service configuration section

### Client solution

The only things you *need* to do.

Things you *may* do, but with the *caveats* that you shouldn’t unless you need to. And you can keep the two teams very separate.

You would then, however, need to have skills in using Angular. For this reason separate manual for configuring and customising the Naked Objects Client.

## Writing your own applications

Define domain types

Define domain functions

Add persistent domain types to DbContext, and specify mapping if required

Register domain types with Naked Functions, manually or reflectively

Register all domain functions with NakedFunctions, manually or reflectively

Register types that define main menu functions, Register all domain functions with NakedFunctions, manually or reflectively

# Defining domain types

Types should be immutable. Can be immutable classes, but, for C#9 recommendation is to use records, because it permits the use of ‘with’. Later versions of C# are likely to eliminate remaining distinctions.

## Properties

Can be of any domain type, or fixed list of value types

use get; init;

All properties must be virtual (on persistent objects

### Attributes for use on properties

DescribedAs

Hidden

Mask

MemberOrder

MultiLine

Named

Versioned (ref also the

## Collections

### Attributes for use on collections

TableView

RenderEagerly

## Overriding default methods common to all .NET types

### Overide the ToString method to define a title for an instance

Building titles using string interpolation

### [Temp] override the GetHashCode and Equals methods

## Class level attributes

Bounded

DescribedAs TODO: review

Named

Plural

# Defining domain functions

### Attributes for use on functions

CreateNew

DescribedAs

Disabled

DisplayAsProperty

Edit

MemberOrder

Named

### Attributes for use on parameters

DefaultValue

DescribedAs

Disabled

Mask

MaxLength

MinLength

MultiLine

Named

# System services & configuration

General introduction to configuring the

## Authentication

## Authorization

## Auditing

## Profiling

## I18N

# Appendices

## Attributes – summary

Table, derived from existing spreadsheet.