OpenAccess ORM

Quick Start Guide for Telerik OpenAccess ORM

Contents

Overview	3
Product Installation	3
Building a Domain Model	5
Database-First (Reverse) Mapping	5
Creating the Project	6
Creating Entities From the Database Schema	6
Model-First (Forward) Mapping	
Creating the Project	
Creating an Entity	12
Creating the Database	16
Updaing the Model and/or the Database (aka Round-trip Mapping)	19
Pulling in Database Schema Changes	
Pushing Domain Model Changes to the Database	24
Using Fluent Mapping API	
Creating the project	
Building the Model	
Deploying the Database	
Working with the OpenAccess Data Model	
Creating the Client Application	
Consuming a Model – CRUD and WCF Services	
Profiling an Application	
Configure The Data Model for Profiling	
Configuring a Fluent Mapping Model for Profiling	
Configuring a Visual Designer Model (Forward or Reverse Mapping) for Profiling	
Real-time Profiling	43
Configuring an Application	
Connecting the Profiler	44
Viewing Offline Profiling Session Logs	



Overview

Getting started with Telerik OpenAccess ORM is quick and easy. Following the **steps** below will ensure it is a smooth process.

- 1. Install Telerik OpenAccess ORM
- 2. Build a Data Model
- 3. Connect the Model to UI

Product Installation

The Telerik installer can be downloaded through your account on <u>Telerik.com</u> or from the <u>OpenAccess ORM page</u> (trial version only).

Once the download completes, unzip the package, and run the installer. The installer will walk you through the installation steps.

The installer will create an OpenAccess Folder under the Telerik folder in the start menu. In this folder there it will add a shortcut to launch the <u>Profiler and Tuning Advisor</u>, as well as a shortcut to the complete documentation and to this document.





Quick Start Guide for Telerik OpenAccess ORM

The installer will also add new project templates to Visual Studio:

New Project	er bes		? ×
Recent Templates	ET Framework 4 Sort by: Default	Search Installed Temp	lates 🔎
Installed Templates	Enable Windows Azure Tools	Visual C#	
✓ Visual C# Windows		A project for creating Windows Forms user	an application with a interface
Web	Activity Library	Visual C#	
Reporting	WCF Workflow Service Application	Visual C#	
Silverlight Silverlight for Windows Phone	RadControls Windows Phone Application	Visual C#	
WCF Workflow	C# Telerik OpenAccess Class Library	Visual C#	
XNA Game Studio 4.0 ◢ Telerik	C# Telerik OpenAccess Fluent Library	Visual C#	
Silverlight Silverlight for Windows Phone	Telerik OpenAccess MVC2 Application	Visual C# ≡	
Test Web	Telerik OpenAccess Web Application	Visual C#	
Vindows ▶ Other Languages	Windows Phone Application	Visual C#	
Other Project Types Database	Windows Phone Databound Application	Visual C#	
Online Templates	Windows Phone Class Library	Visual C# 🕌	
Name: WindowsForms	cation1		
Location: D:\Projects\Exar	1	Browse	
Solution name: WindowsForms	cation1	Create directory for so Add to source control	blution I
		0	K Cancel

New Web Site			8 X
Recent Templates	.NET Framework 4 Sort by: Default		Search Installed Templates
Installed Templates Visual Basic	ASP.NET Web Site	Visual C#	Type: Visual C# A project for creating an web site using
Visual C# Online Templates	ASP.NET Web Site (Razor v2)	Visual C#	ASP.NET web site and Telerik OpenAccess ORM.
	ASP.NET Web Site (Razor)	Visual C#	
	ASP.NET Empty Web Site	Visual C#	
	ASP.NET Dynamic Data Entities Web Site	Visual C#	
	ASP.NET Dynamic Data Ling to SQL Web Site	Visual C#	
	CH WCF Service	Visual C#	
	ASP.NET Reports Web Site	Visual C#	
	Telerik OpenAccess WebSite	Visual C#	
	ASP.NET Crystal Reports Web Site	Visual C#	
Web location: File System	C:\Users\holt.TELERIK\Documents\Visual Studio 2010\WebSites	\WebSite ▼	Browse
			OK Cancel



What Do These Templates Do?

Telerik OpenAccess Class – Creates a new class library project with an OpenAccess domain model built using the visual designer and <u>forward</u> or <u>reverse</u> mapping.

Telerik OpenAccess Fluent Library - Creates a new project using the OpenAccess ORM <u>Fluent</u> <u>Mapping API</u>.

Telerik OpenAccess MVC2 Application – Creates a new solution with an MVC 2 web project, and a Telerik OpenAccess Fluent Library project.

Telerik OpenAccess Web Application – Creates a new ASP.Net AJAX web project, with a Telerik OpenAccess Domain model built using the visual designer and <u>forward</u> or <u>reverse</u> mapping.

Next Steps...

Now that OpenAccess is installed, the next step is to build a Domain Model

Building a Domain Model

Telerik OpenAccess ORM makes it very simple to start working with data. The first choice you need to make is how you would like to configure the mapping:

- If you have an existing database, and would like to create entities from the tables, views, or stored procedures, <u>Database-First (Reverse) Mapping</u> will be the best path.
- If you do not have an existing database, and would like to build your classes first, and have the database generated automatically, then <u>Model-First (Forward) Mapping</u> will be the best path for you.
- If you would like complete control over the code in your entities, databases updates, and data mapping, then the <u>Fluent Mapping API</u> will be the best path for you.

Database-First (Reverse) Mapping

OpenAccess ORM reverse mapping allows developers to map an existing database schema to .Net objects, which can be utilized by any .Net application.

Note

For this section the guide will use a MS SQL database named **OpenAccessQuickStartDB**. To create this example database run the following script:

```
USE [master]
GO
CREATE DATABASE [OpenAccessQuickStartDB] ON PRIMARY
( NAME = N'OpenAccessQuickStartDB', FILENAME = N'C:\Program
Files\Microsoft SQL
Server\MSSQL10.SQLEXPRESS\MSSQL\DATA\OpenAccessQuickStartDB.mdf', SIZE
= 2048KB , MAXSIZE = UNLIMITED, FILEGROWTH = 1024KB )
LOG ON
```



```
( NAME = N'OpenAccessQuickStartDB_log', FILENAME = N'C:\Program
Files\Microsoft SQL
Server\MSSQL10.SQLEXPRESS\MSSQL\DATA\OpenAccessQuickStartDB_log.ldf',
SIZE = 1024KB , MAXSIZE = 2048GB , FILEGROWTH = 10%)
GO
USE [OpenAccessQuickStartDB]
GO
CREATE TABLE [dbo].[Customer](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [Name] [varchar](255) NULL,
    [DateCreated] [datetime] NULL,
    [EmailAddress] [varchar](255) NULL,
CONSTRAINT [PK_Customer] PRIMARY KEY ([Id]))
GO
```

Creating the Project

To create a new Telerik OpenAccess domain model using reverse mapping we will need to create a project first:

- 1. Select File > New Project in Visual Studio
- 2. Select Visual C# / Visual Basic in the Installed Template tree.
- 3. Then select **Telerik OpenAccess Class Library**. Name the project **QuickStartEntities**, and then click **OK**.



This will envoke the New Domain Model Wizard for

Creating Entities From the Database Schema



In the Telerik OpenAccess New Domain Model Wizard select Populate from Database. Then select the target database, name the model QuickStartEntities, and click Next.

📕 Telerik OpenAccess	ess New Domain Model Wizard	
Select OpenA	Access Domain Model Type	ORM
Choose OpenAco	ccess domain model type	
Populate from database	Empty domain model	
This option will let y	t you create a domain model based on an existing database schema (schema first).	
Backend		
	Backend: Microsoft SQL Server	
Model		
,	Model Name: QuickStartEntities	
Model	el Namespace: Use the Project Namespace: QuickStartEntities 	
	Use Custom Defined EntitiesModel	
	< Previous Next > Finish	Cancel

2. Add or select an existing connection, name the connection string QuickStartDBConnection, and then click Next.



📕 Telerik OpenAccess New Domai	Model Wizard	
Setup Database Conn Choose the database you want t	ection ^{o map}	ORM
Choose Connection		
Connection:	\sqlexpress.OpenAccessQuickStartDB.dbo	add New Connection
Set Connection Manually		
Connection String:	Data Source=.\sqlexpress;Initial Catalog=OpenAccessQuickStartDB;Integrated Security=Tru	Je
Connection String Name:	QuickStartDBConnection	
	< Previous Next > Fin	ish Cancel

3. In the next screen, select the **tables**, **views**, and **stored procedures** to import into the domain model, for this guide the **Customer** table will be imported. Next, Click **Finish.**



Telerik OpenAccess New Domain Model Wizard	
Choose Database Items All selected items will be included in the data model	ORM
Choose Schemas	Check All
☑ dbo	
Selected schemas: dbo	
OpenAccessQuickStartDB	
✓ Tables ✓ Views ✓ Stored Procedures Expand	All Collapse All
Image: Customer Image: Custome	
🔲 Group by schemas	
Use schema names for namespaces	
< Previous Next > Finish	Cancel

When you click Finish , Telerik OpenAccess ORM will add two references to your project:



It will also add an app.config, which stores the connection string, and an .rlinq file, which stores information about the mapping.



灵 Solution 'QuickStartEntities' (1 project)
QuickStartEntities
Properties
A Comparison Comparison Activity of Compar
- Microsoft. CSharp
- System
- System.Core
- System.Data
- System.Data.DataSetExtensions
- System.Xml
- System.Xml.Linq
- Telerik.OpenAccess
Telerik.OpenAccess.35.Extensions
🚯 App.Config
▲ La EntitiesModel.rlinq
Customer.generated.cs
EntitiesModel.cs
🖹 EntitiesModel.rlinq.diagram

Next Steps...

At this point the model is ready to be <u>used in an application</u>, or run the Data Service Wizard to create a service layer over the model.

Model-First (Forward) Mapping

OpenAccess ORM forward mapping allows developers to concentrate on building classes, and will automatically generate a database based on those classes.

Creating the Project

To create a new Telerik OpenAccess domain model using forward mapping:

- 1. Select File > New Project in Visual Studio
- 2. Select Visual C# or Visual Basic in the **Installed Template** tree.
- 3. Then select **Telerik OpenAccess Class Library**. Name the project **OpenAccessQuickStartModel**, and then click **OK**.





4. In the **Telerik OpenAccess New Domain Model Wizard** select **Empty Domain Model**. Then select the target database, name the model **QuickStartEntities**, and click **Finish**.

Telerik OpenAccess New Domain Model Wizard	_ 0	23
Select OpenAccess Domain Model Type	0	RM
Choose OpenAccess domain model type		
Populate Empty from domain database model		
Choose this option if you want to start with a domain model that will be used to create the database schema later (model first). Backend Backend: Microsoft SQL Server		
Model		
Model Name: QuickStartEntities		
Model Namespace: Our Use the Project Namespace: OpenAccessQuickStartModel		
C Use Custom Defined EntitiesModel		
< Previous Next > Finish	Cancel	

When you click finish, the wizard will enhance the project to work with OpenAccess. In addition, it will add the required references, and an .rlinq file to the project.



Quick Start Guide for Telerik OpenAccess ORM

- Solution 'OpenAccessQuickStartModel' (1 project)
- OpenAccessQuickStartModel
 - Properties
 - A Beferences
 - Microsoft. CSharp
 - System
 - System.Core
 - System.Data
 - System.Data.DataSetExtensions
 - System.Xml
 - System.Xml.Ling
 - Telerik.OpenAccess
 - Telerik.OpenAccess.35.Extensions
 - 🛛 🖺 EntitiesModel.rling
 - EntitiesModel.cs
 - 嶜 EntitiesModel.rlinq.diagram

Creating an Entity

OpenAccess provides new toolbox items to make model creation a simple drag and drop operation.

1. Drag a new **Domain Class** out of the toolbox, and **drop** it on the design surface.



2. **Double click** on the entity title, which will enable you to edit the entity's name. Name the entity **Customer**.





3. Add properties to an entity by **right clicking** on the **properties** section, and selecting **Add new Property**.

🕰 Customer	۸		
Properties			
Implementation		Collapse	
+ Navigation		Add new Property	
- Navigation		Diagram Options	•
		Show Model Settings	
		Update From Database	
		Update Database From Model	
	P.	Properties	Alt+Enter

Add the following Properties to the Customer entity:

- Id : int
- Name : string
- DateCreated : DateTime
- EmailAddress : string

The entity should now look like this:





- 4. Next we need to configure the primary key, and set which property will be used as the identity mechanism.
- 5. With the **Customer** entity selected, expand the Properties window.
- In the Properties window, Select the Identity Mechanism dropdown, and choose DatabaseServerCalculated. This tells OpenAccess that the value of the Entities identity field will set by the database server.

Properties		▼-₽X
Customer Domain Cla	ass	•
8: 2 0		
Access Modifier		Public
Cache Policy		Default
Concurrency Mem	nber	r (AUTO)
Concurrency Mod	e	Default
Description		
Fill Color		253, 184, 119
Identity Mechanis	m	atabaseServerCalculated ▼
Inheritance Modif	Defa	fault
Kind	Data	tabaseServerCalculated
Name	Guio	id
Namespace	Higl	ghLow
Outline Color		213, /1, 0
Text Color		Black
Update Schema		False
Identity Mechanism Description for Telerik Mechanism	.Dat	ta.Dsl.DomainClass.Identity

For more information about the other Identity Mechanisms supported by OpenAccess please read this <u>documentation article</u>.



 Now we need to specify which property will be used as the entiti's Identity. To do this, select the Id property of the Customer entity. Expand the Properties window, and set the Identity property to True.

	Properties	* -⊐ X
	Id Property	•
Customer 🛞	₽ <mark>2↓</mark> 🖸	
	Access Modifier	Public
Properties	Description Field Name	
🞦 Id : int	Identity	True 🔻
Name : string	Inheritance Modifier	True
DateCreated : DateTime	Kind	False
EmailAddress : string	Load Behavior	Derault
Implementation	Name	Id
Navigation	Nullable	False
	Туре	int
III	Identity Gets or sets whether the p mechanism.	property is part of an identity

8. The last step is to map the **Customer** entity to a table in the **database**. OpenAccess provides a mechanism to automatically create this mapping. To use this feature, expand the **Mapping Details Editor**.





Note

If the mapping details tab is not visible as indicated in the screenshot, access through the Visual Studio Menu, by navigating to: **View>Other Windows>Model Details Explorer**.

Solution Explorer Team Explorer Server Explorer Anchitecture Explorer Call Hierarchy Code Definition Window Object Browser Error List Output Start Page Task List	Ctrl+W, S Ctrl+W, M Ctrl+W, L Ctrl+W, N Ctrl+W, K Ctrl+W, C Ctrl+W, C Ctrl+W, D Ctrl+W, D Ctrl+W, D		Det	ug • 🕼 ind	ex.html.14	- 123 G 3× 8 8
Team Explorer Server Explorer Architecture Explorer Call Hierarchy Class View Code Definition Window Object Browser Error List Outpot Start Page Task List	Ctrl+W, M Ctrl+W, L Ctrl+W, N Ctrl+W, K Ctrl+W, C Ctrl+W, C Ctrl+W, D Ctrl+W, J Ctrl+W, E Ctrl+W, O		_			_
Server Explorer Architecture Explorer Call Hierarchy Calls Definition Window Object Browser Error List Output Start Page Task List	Ctri+W, L Ctri+W, N Ctri+W, K Ctri+W, C Ctri+W, D Ctri+W, J Ctri+W, E Ctri+W, O					
Architecture Explorer Call Hierarchy Class View Code Definition Window Object Browser Error List Outpot Start Page Task List	Ctri+W, N Ctri+W, K Ctri+W, C Ctri+W, D Ctri+W, J Ctri+W, E Ctri+W, O					
Call Hierarchy Class View Code Definition Window Object Browser Error List Output Start Page Task List	Cbrl+W, K Cbrl+W, C Cbrl+W, D Cbrl+W, J Cbrl+W, E Cbrl+W, O					
Class View Code Definition Window Object Browser Error List Output Start Page Task List	Ctrl+W, C Ctrl+W, D Ctrl+W, J Ctrl+W, E Ctrl+W, O					
Code Definition Window Object Browser Error List Output Start Page Task List	Ctrl+W, D Ctrl+W, J Ctrl+W, E Ctrl+W, O					
Object Browser Error List Output Start Page Task List	Ctrl+W, J Ctrl+W, E Ctrl+W, O					
Error List Output Start Page Task List	Ctrl+W, E Ctrl+W, O					
Output Start Page Task List	Ctrl+W, O					
Start Page Task List						
Task List						
	Ctrl+W, T			Customer	(8)	
Toolbox	Ctrl+W, X					
Find Results				Properties		
Other Windows		2	Command Window	Ctrl+W, A		
Toolbars	,		Web Browser	Ctrl+W, W		
Full Screen	Shift+Alt+Enter	9	Laver Explorer			
Navigate Backward	Ctrl+-	1.00	Macro Explorer	Alt-FB		
Navigate Forward	Ctrl+Shift+-	10	Source Control Explorer			
Next Task		5	UML Model Explorer	Ctrl+ Ctrl+U		
Previous Task			Bookmark Window	Ctrl+W B		
Properties Window	Ctrl+W P	54	Package Manager Console	contra e		
Property Pages	Shift+F4	-	OpenAccess Model Explorer			
		122	Entity Diagrams Details Editor			
		12	Entity Diagrams Schema Explorer			
		6	Document Outline	Ctrl+W, U		
		64	History			
	Find Beauts Coher Windows Coher Windows Coher Windows Full Screen Norigate Forward Nori Task Properties Undow Property Pages.	Find Result: Cher Windows Cher Windows Cher Windows Full Screen Shift-Alt-Enter Norigate Forward Chi- Norigate Forward Shift-Alt-Enter Shift-Alt-Enter Chi- Norigate Shift-Alt-Enter Shift-Alt-	Find Results Cher Windows Cher Windows Find Screen Shift-Alt-Enter Find Screen Shift-Alt-Enter Find Screen Cher Cher Norigate Forward Cher Norigate Forward Cher Proving Task Properties Window Cher W, P Property Pages Find Cher Find Cher Find Cher Find Cher Find	Find Besults Coher Windows Coher Windows Coher Window Co	Find Besults • • Procerties Other Windows • • • • Procerties Other Windows •	Find Bealds Cher Windows Cher Window Che

9. With the Mapping Details Editor expanded, check **Use Default Mapping**.

Mapp	ing Details Editor - Table Mappings for	Customer		×⊣×
	Mapped to: customer		✓ Use Default Mapping	
ൽ	Column	Operation	ClassProperty	
1	📁 id : int	< Mapping >	🖺 ld : Int32	
	🔳 nme : varchar	< Mapping >	🕅 Name : String	
	date_created : datetime	< Mapping >	DateCreated : DateTime	
	email_address : varchar	< Mapping >	🖾 EmailAddress : String	
<u>е</u> п.	sit Tast Window 📑 JustCodo Errors 🚔	Error List 🔽 Common	d Window 🚝 Immediate Window 📮 Outout 👿 Manajag D	stails Editor

What Just Happened?

When using the **Use Default Mapping** option, OpenAccess will automatically build a database table schema for an entity based on its properties, and their data types. To customize this mapping please read this <u>documentation article</u>.

With the domain model built, the next step is to <u>create the database</u> the domain model will use to save information.

Creating the Database

OpenAccess will generate database creation, or migration, scripts based on the changes made in the designer. In addition, it can execute these scripts, making database creation, and migration very simple.

- 1. Save the OpenAccess Domain model by pressing Ctrl+s.
- 2. Right click on the design surface, and select Update Database from Model.



🕰 Customer 🛞			
 Properties Id : int Name : string DateCreated : DateTime EmailAddress : string 		Add Diagram Options Show Model Settings	,
Implementation		Update Database From Model	
Navigation		Validate All	
	P.	Properties	Alt+Enter

3. In the wizard, **Add** or **select** an existing database connection. Name the connectionstring **QuickStartDBConnection**, and then click **Next**.

Update Database from Model	
Setup Database Connection Choose the database you want to map	ORM
Choose Connection	
Connection: \sqlexpress.OpenAccessQuickStartDB.dbo	Add New Connection
Set Connection Manually	
Connection String: Data Source=.\sqlexpress;Initial Catalog=OpenAccessQuickStartDB;Integrated	I Security=True
Connection String Name: QuickStartDBConnection	
< Previous Next >	Finish Cancel

4. Ensure **Create Database** is the selected **Update Option**, and click **Next**.

Update Database from Model Select changes page		B				
Update Options Create Database Migrate Database	your database schema.					
Model Name			Select all	Deselect all	Expand all	Collapse all
From model	Operation	To	database	New value	Old va	lue
Tables	Modify	Tables				
 customer 	Add					
FullName	Add	FullName		'customer'	<empty></empty>	
Name	Add	Name		customer	<empty></empty>	
 Columns 	Modify	Columns				
I date_created	Add					
email_address	Add					
D 🔳 id	Add					
▷ 🔳 nme	Add					
			< Previous	Next >	Finish	Cancel

5. Select Create Script File and Execute, and click Finish.



Update Database from Model					
Summary page					ORM
Set the deployment options. Pr	eview and execute the generated sql so	ript.			
Summary Options					
Create Script File	Oreate Script File and Execute				
Deployment options					
Destination Folde	r			Browse	
Add Prefi	x				
	Append Date-Time Stamp				
Example	QuickStartGuideConnection(2011-	12-07 03-09-44).rlinq.sql			
Script to be applied to the	database				
				C	Copy to Clipboard
OpenAccessQuickStartMode	el.Customer				
[customer_id] int NOT NULL	, <internal-pk></internal-pk>				=
[nme] varchar(255) NULL,	name				
[email_address] varchar(255) NULL,emailAddress				
[date_created] datetime NO	T NULL,dateCreated				
CONSTRAINT [pk_customer]	PRIMARY KEY ([customer_id])				*
		< Previous	Next >	Finish	Cancel

- 6. After clicking finish, OpenAccess will save a copy of the script, and execute it against the database.
- 7. The database should look like this:

-	OpenAccessQuickStartDB
	🕀 🧰 Database Diagrams
	🖃 🧰 Tables
	표 🚞 System Tables
	🖃 🧾 dbo.customer
	🖃 🚞 Columns
	📃 nme (varchar(255), null)
	🦞 id (PK, int, not null)
	email_address (varchar(255), null)
	date_created (datetime, not null)

Next Steps...

At this point the database, and model are ready to be <u>used in an application</u>., or run the Data Service Wizard to create a service layer over the model.

Updaing the Model and/or the Database (aka Round-trip Mapping)

Round trip mapping is built into the OpenAccess Visual Designer. It allows developers to make changes in the database schema, and pull those into their domain model, or update their



domain model, and push those changes to the database schema. To leverage round trip mapping, start with either **Forward Mapping**, or **Reverse Mapping**. Once you have a model in place, you are free to push and pull changes as needed.

Pulling in Database Schema Changes

Note

For this example, a column named DOB was added to the Customer table. Use the following script to create the new column in the database:

```
ALTER TABLE dbo.Customer ADD
DOB datetime NULL
GO
```

1. **Right click** on the visual designer surface, and click **Update from Database**.

🐱 QuickStartEntities - Microsoft Visual Studio (Ad	dministrator)		- 0 X
File Edit View Telerik JustCode Project	Build Debug Team Data For	mat Tools Architecture Test Ana	ilyze Window
Help : 🔁 • 🔛 • 😂 🛃 🥔 🐰 🖦 📇 🧐 • 🤉 : Collapse All Expand All 🏢 🎦 🔂 😂 🚇	■ - ,	v Debug v 🖡	≝ <u></u>
Toolbox · 무 × EntitiesMod	del.rling ×		<u> </u>
▲ Designer Toolbox			<u>∧</u>
Pointer Association Comment Comment Link	Customer 🛞		del Schema Ex
A Domain Class			p p p
i _{p₀} Inheritance ⊷ Interface	 Properties Id : Int32 		rer 😰 Op
General There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.	 DateCreated : D EmailAddress : S Name : String Implementation Navigation 	Add Diagram Options Show Model Settings Update From Database Update Database From Model Validate All Properties	Alt+Enter
糦 Server Explorer 🔆 Toolbox 🔍	m		perties 💐
📓 JustCode Errors 📸 Error List 🔳 Output 📑	Pending Changes 📴 Test Results	📌 Package Manager Console 🧮 Code	e Metrics Resu
Violerrors in solution Creating project	QuickstartEntities project creation	i successful.	

 In the first screen, check each table, view, or stored procedure that should be imported or updated in the domain model, and click Next. In this example the Customer entity will be updated.



Update From Database		_ 🗆 X
Choose Database Items All selected items will be included in the data model		ORM
Choose Schemas		Check All
Selected schemas: dbo		
✓ Tables ✓ Stored Procedures ✓ □ Tables ✓ □ Customer □ ☑ Views □ ☑ Stored Procedures	Expand All	Collapse All
Group by schemas Use schema names for namespaces		
< Previous Next >	Finish	Cancel

3. On the next screen, **select** which changes you would like to **apply** to the domain model, and click **Next**.



Ipdate From Database	(1) (m - 1) (1	helter) # 5		
elect which changes to apply to	your model.			
			Select all Deselect all	Expand all Collapse all
From database	Operation	To model	New value	Old value
🖌 📝 💼 Tables	Modify	Tables		
🔺 📝 🧾 Customer	Modify	Customer		
 Columns 	Modify	Columns		
🔺 📝 🧮 DOB	Add			
AdoType	Add	AdoType	93	<null></null>
✓ IsNullable	Add	IsNullable	True	<null></null>
Name	Modify	Name	DOB	<null></null>
SqlType	Add	SqIType	datetime	<null></null>
		< Previo	us Next >	Finish Cancel

Important Note: Not all changes in the database schema have to be pushed over to the domain model. Only push changes OpenAccess should know about.

4. On the final screen, review the changes that will be applied to the domain model, and click **Finish**.



review selected chan	ges			OR
From database	Operation	To model	New value	Old value
🔹 🛅 Tables	Modify	Tables		
Customer	Modify	Customer		
 Columns 	Modify	Columns		
DOB	Add			
AdoType	Add	AdoType	93	<null></null>
IsNullable	Add	IsNullable	True	<null></null>
Name	Modify	Name	DOB	<null></null>
SqIType	Add	SqIType	datetime	<null></null>
		L D I		

 When you click finish the domain model objects will be updated based on the selections made in the **Update from Database** wizard. In this case, a DateTime property named **DOB** was added to the **Customer** entity.



👓 Quick	StartEntities - Microsoft Visu	Studio (Administrator)	Of Street West				×
File Ed	lit View Telerik JustCod	Project Build Debug Team D	ata Format	Tools Architecture	Test Analyze	Window	w
Help : 🔂 🕶 : Collap	🛅 🕶 🚅 🛃 🧊 🐰 斗 🕻 pse All 🛛 Expand All 🔠 🏥	♥) • (° - ,⊒ • ⊑,) 2 🚂 🚔 €, ⊙, 100% • ₌ ;	New Work Iter	- Debug n - 🛅 → 🖄 -	• 🖄 👻		
Toolbox	. - 4 х	ntitiesModel.rlinq* ×				-	80
⊿ Desig	gner Toolbox					-	ş
k	Pointer						del s
Ľ.	Association						Sche
<u> </u>	Comment	(4)				=	ma
L.,	Comment Link	📽 Customer 🖄					Expl
1	Domain Class						orer
۴ ۵	Inheritance	Properties					
	Interface	21 Id : Int32					ę
✓ Gene There this g this te	eral e are no usable controls in group. Drag an item onto ext to add it to the toolbox.	DateCreated : D EmailAddress : S Name : String DOB : DateTime Implementation Navigation					enAccess Model Explorer 💣 Proper
🚉 Serv	ver Explorer 🔆 Toolbox Code Errors 豫 Error List 🔳	III Dutput 🛐 Pending Changes 🧮 Tes	t Results 📌 P	lackage Manager Consi	ole 🧮 Code Met	rics Resu	ties 🏹
🕜 No e	rrors in solution Cre	ng project 'QuickStartEntities' projec	t creation suc	cessful.			

Pushing Domain Model Changes to the Database

1. Add a new property string named **PhoneNumber** to the **Customer** entity.



- 2. Save all pending changes to the domain model by pressing Ctrl + S.
- 3. Right Click the design surface, and select Update Database from Model.





 Select Migrate Database, check each change that should be applied to the database, and click Next. In this case,



Update Database from Model		1.000		
Select changes page Select which changes to apply to your	r database schema.			ORM
Update Options				
Create Database				
Migrate Database				
Inigrate Database				
Model Name				
		Select all	Deselect all Expand a	II Collapse all
From model	Operation	To database	New value	Old value
▲ ▼ ables	Modify	Tables		
🔺 📝 🧾 Customer	Modify	Customer		
 Columns 	Modify	Columns		
🖻 🗹 🧾 PhoneNumber	Add			
		< Previous	Next > Fini	sh Cancel

5. Select Create script file and execute, and then click Finish.



Update Database from Model	
Summary page Set the deployment options. Preview and execute the generated sql script.	ORM
Summary Options	
Create Script File Create Script File and Execute	
Deployment options	
Destination Folder Add Prefix I Append Date-Time Stamp	Browse
Example: QuickStartDBConnection(2011-12-07 05-25-36).rlinq.sql	
Script to be applied to the database	
	Copy to Clipboard
add column for field _phoneNumber ALTER TABLE [Customer] ADD [phone_number] varchar(255) NULL go	
< Previous	Next > Finish Cancel

6. At this point OpenAccess will create, and save, a copy of the script in the specified destination folder, and it will execute the migration script against the database. Once complete the database should look like the following:

🖃 🔳 (dbo.Customer
-------	--------------

🖃 🚞 Columns

- 💡 Id (PK, int, not null)
- Name (varchar(255), null)
- DateCreated (datetime, null)
- EmailAddress (varchar(255), null)
- DOB (datetime, null)
- phone_number (varchar(255), null)

Note

In this example the column name was not specified for the PhoneNumber property. In this case OpenAccess will automatically create a column name based on the property name. To learn how to configure the column name, please read this <u>documentation</u> <u>article</u>.

Using Fluent Mapping API

OpenAccess provides a fluent mapping api, also known as code-first, for defining data model models using only code. The fluent mapping api gives developers complete control over the domain model mapping configuration, and schema management.



Creating the project

To create a new Telerik OpenAccess domain model using the Fluent Mapping API:

- 1. Select File > New Project in Visual Studio
- 2. Select Visual C# or Visual Basic in the Installed Template tree.
- 3. Then select **Telerik OpenAccess Fluent Library**. Name the project **QuickStartEntities**, and then click **OK**.



4. At this point Telerik OpenAccess will create a new project with the required references, and add a basic template for a fluent mapping project.



- Solution 'QuickStartEntities' (1 project)
- QuickStartEntities
 - Properties
 - A Contraction Contraction Contraction
 A Contraction Contraction
 A Contractio
 - Microsoft. CSharp
 - 🗉 System
 - System.Core
 - System.Data
 - System.Data.DataSetExtensions
 - System.Xml
 - System.Xml.Linq
 - Telerik.OpenAccess - Telerik.OpenAccess.35.Extensions
 - App.config
 - Product.cs
 - QuickStartEntitiesContext.cs
 - QuickStartEntitiesMetadataSource.cs

What are these files?

The app.config contains the connection string, QuickStartEntityContext.cs defines an OpenAccessContext class, which is the primary way developers access the domain model in an application. QuickStartEntitiesMetadataSource.cs defines a FluentMetadataSource class, which is used to specify object to schema mappings. Product.cs is a simple example POCO object.

Building the Model

- 1. **Expand** the solution explorer.
- 2. **Rename Product.cs** to **Customer.cs**, open the file, and **replace** the product class definition with the code below:

```
public class Customer
{
    public int ID { get; set; }
    public string Name { get; set; }
    public DateTime DateCreated { get; set; }
    public string EmailAddress { get; set; }
}
```

What Does this Code Do?

This code creates a simple customer object that can be mapped to a database table by OpenAccess.

- 3. Next, **Open QuickStartEntitiesMetadataSource.cs**, and **remove** the existing code in the **PrepareMapping** method.
- 4. **Paste** the following code into the **PrepareMapping** method to configure the mapping for the customer object:



```
var configurations = new List<MappingConfiguration>();
var customerMapping = new MappingConfiguration<Customer>();
//map the customer class to a table named customer
customerMapping.MapType(customer => new
{
    ID = customer.ID,
    Name = customer.Name,
    EmailAddress = customer.EmailAddress,
    DateCreated = customer.DateCreated
}).ToTable("Customer");
//Specify the id property for the entity
customerMapping.HasProperty(x => x.ID)
                .IsIdentity(KeyGenerator.Autoinc);
configurations.Add(customerMapping);
```

return configurations;

What Does this Code Do?

This code passes an anonymous type to the MapType method. The properties of the anonymous type will be what OpenAccess uses to create columns in the database. The ToTable method tells OpenAccess what the customer table should be named in the database. Next the entity's Identity field is specified using the IsIdentity() method. KeyGenerator.Autoinc tells OpenAccess that the identity will be calculated by the database.

- 5. Open **QuickStartEntitiesContext.cs**, **remove** the property exposing the **Products** collection.
- 6. **Paste** in the following code to expose the customer collection.

```
public IQueryable<Customer> Customers
{
    get
    {
        return this.GetAll<Customer>();
    }
}
```

This code exposes the customer objects as an IQueryable < Customer > this allows OpenAccess to offload queries to the database to do the heavy lifting. It also makes it easy to query entity sets using standard LINQ.

7. At this point the model is fully configured, next the database needs to be created.



Deploying the Database

OpenAccess can generate database creation, or migration, scripts based on changes made in the QuickStartEntitiesMetadataSource.cs schema mapping file. In addition, it can execute these scripts, making database creation, and migration very simple.

To create or update the database based on the fluent mapping configuration:

1. Create a new Console Application, name it QuickStartBootstrapper, and click OK.



2. Right click on the **Console Application**, and select **Set as StartUp Project**



Solu	tion	Explo	rer	→ ₽
Ŀ,		ء و	84	
~	So	lution	'QuickStartEntities' (2 projects)	
⊿	đ	Quic	kStartBootstrapper	
	⊳		Build	
	Þ		Rebuild	
4	¢.		Repeat Test Run	
	⊳		Clean	
	⊳	2	Publish	
			Run Code Analysis	
			Calculate Code Metrics	
			Project Dependencies	
			Project Build Order	
			Add	•
			Add Reference	
			Add Service Reference	
		1	Manage NuGet Packages	
		æ,	View Class Diagram	
			Set as StartUp Project	
			Debug	•
		1	Add Solution to Source Control	
			Go To Reflector	
		¥	Cut	Ctrl+X
			Paste	Ctrl+V

- 3. Expand the **Solution Explorer**, right click on **References**, and select **Add Reference**.
- 4. Add References to:
 - Telerik.OpenAccess.dll
 - Telerik.OpenAccess.35.Extenstions.dll
 - The model project, in the case of this guide: QuickStartEntites
- 5. Copy the **app.config** from the **QuickStartEnities** project to the QuickStartBootstrapper project.
- 6. Open **Program.cs** in the **Console Application.**
- 7. **Replace** all of the existing code with the following:

```
using System;
using System.Linq;
using QuickStartEntities;
using Telerik.OpenAccess;
namespace QuickStartBootstrapper
{
   class Program
    {
        static void Main(string[] args)
        {
            UpdateDatabase();
```



```
Console.Write("Database update complete! Press any key to
  close.");
        Console.ReadKey();
    }
   private static void UpdateDatabase()
   {
        using (var context = new QuickStartEntitiesContext())
        {
            var schemaHandler = context.GetSchemaHandler();
            EnsureDB(schemaHandler);
        }
    }
   private static void EnsureDB(ISchemaHandler schemaHandler)
    {
        string script = null;
        if (schemaHandler.DatabaseExists())
        {
            script = schemaHandler.CreateUpdateDDLScript(null);
        }
        else
        {
            schemaHandler.CreateDatabase();
            script = schemaHandler.CreateDDLScript();
        }
        if (!string.IsNullOrEmpty(script))
        {
            schemaHandler.ExecuteDDLScript(script);
        }
    }
}
```

What Does this Code Do?

The first thing this code does is create a new instance of the **QuickStartEntitiesContext**, it is wrapped in a using statement to make sure it is properly disposed. Next it checks to see if the database exists. If it does not exist, the database is created, and then the schema is applied. If the database already exists, OpenAccess will create and run a migration script against the database.

At this point when you run the console application OpenAccess will update the database based on any changes made in the **QuickStartEntitiesMetadatasource** class.

Next Steps ...

}



At this point the model is ready to be <u>used in an application</u>, or run the Data Service Wizard to create a service layer over the model.

Working with the OpenAccess Data Model

OpenAccess data models can be used with just about any .NET platform. The model can be used directly in web, and windows application, and for rich client.

Creating the Client Application

Note

For this guide we will create a basic console application to interact with the OpenAccessContext; however, the same concept apply regardless of the application platform(WinForms, WPF, MVC, ASP.NET, etc.) being used . For more in-depth guides about using OpenAccess on other platforms please refer to the <u>OpenAccess ORM SDK</u>, or this <u>documentation article</u>.

1. **Create** a new **Console Application** in the same solution as the <u>previously built</u> <u>OpenAcccess Data Model</u>, give the application a name, and click **OK.**



- 2. Right click on the Console Application, and select Set as StartUp Project
- 3. Expand the Solution Explorer, right click on References, and select Add Reference.
- 4. Add References to:
 - Telerik.OpenAccess.dll
 - Telerik.OpenAccess.35.Extenstions.dll



- The QuickStartEntities project containing OpenAccess data model created using this guide
- 5. Copy, and paste the **app.config** from the **QuickStartEntities** project to the **Console Application**.
- 6. Open **Program.cs** in the **Console Application.**
- 7. **Replace** the existing code with the following:

```
using System;
using System.Linq;
using QuickStartEntities;
namespace QuickStartUI
{
    class Program
    ł
        static void Main(string[] args)
        {
            using (var dbContext = new QuickStartEntitiesContext())
            {
                // Add a new Customer.
                Customer newCustomer = new Customer();
                newCustomer.Name = "New Customer";
                newCustomer.DateCreated = DateTime.Now;
                dbContext.Add(newCustomer);
                // Add another new Customer.
                Customer newCustomer2 = new Customer();
                newCustomer2.Name = "New Customer 2";
                newCustomer2.DateCreated = DateTime.Now;
                dbContext.Add(newCustomer2);
                // Commit new customers to the database.
                dbContext.SaveChanges();
                // Get the first Customer using LINQ and modify it.
                Customer firstCustomer =
dbContext.Customers.FirstOrDefault();
                firstCustomer.Name = firstCustomer.Name + " Updated";
                // Commit changes to the database.
                dbContext.SaveChanges();
                // Use LINQ to retrieve Customer with name 'New
Customer'.
                Customer customerToDelete = (from c in
dbContext.Customers
                                              where c.Name == "New
Customer 2"
                                              select
c).FirstOrDefault();
```



```
// Delete the 'New Customer' from the database.
    dbContext.Delete(customerToDelete);
    // Commit changes to the database.
    dbContext.SaveChanges();
    }
    Console.Write("All changes executed properly, press any key
    to close.");
        Console.ReadKey();
    }
}
```

What Does this Code Do?

The first thing it does is create an instance of the **OpenAccessContext** created early in this guide. The context is created in a using statement to make sure it is properly disposed of after it is used. The next two section of code add two new customers to the OpenAccess context, and then saves them to the database. Next, LINQ is used to retrieve the first customer record, and modify its name. Those changes are then sent to the database using the SaveChanges method on the OpenAccessContext. The last section retrieves a customer by its name using LINQ, and then uses the Delete method on the OpenAccessContext to remove that record from the database.

8. At this point the application is ready to run. When it runs, it will add two new customers to the database, modify one, and then delete one. After running the application there should be one customer entry in the database named "New Customer_Updated."

At this point the data model has been built, an application has been created to interact with the data model, and changes to objects can be persisted to the database. The next step is to expand the data model, and UI application to meet your needs.

Consuming a Model – CRUD and WCF Services

Please, check the following help topics on how to use the created OpenAccess ORM model in the application(s) that you are working on:

- Consuming a Model CRUD
- Using OpenAccess with WCF Services

Next Steps ...

Once the model and application building is complete, hook up the **OpenAccess Profiler and Tuning Advisor** and <u>check performance</u>, or look for potential issues in the data model using the profiler's built in alert system.



Profiling an Application

The **OpenAccess Profiler and Tuning Advisor** makes it easy for developers to see how OpenAccess is working behind the scenes. All of the SQL queries sent to the database server, and the LINQ statement that generated them are easily browsable in the profiler. In addition, the profiler has a built in alert system which will notify developers about potential issues in the domain model, and suggest resolutions.

Configure The Data Model for Profiling

Configuring a Fluent Mapping Model for Profiling

- 1. Create a fluent mapped data model using the <u>previous section</u> in this guide.
- 2. Add a reference to **Telerik.OpenAccess.ServiceHost.dll** in the project that will consume the domain model.
- 3. Open the OpenAccessContext class, if created from this guide, it is the file named **QuickStartEntitiesContext.cs**.
- 4. **Replace** the existing backendConfiguration declaration with the following depending on the desired method of profiling:

Real-time Profiling

In Real-time profiling, OpenAccess reports its events, and metrics directly to the profiler via a service.

```
private static BackendConfiguration GetConfiguration()
{
```

BackendConfiguration backendConfiguration = new BackendConfiguration()

```
{
    Backend = "mssql"
};
// defines the size of the metric store in memory
backendConfiguration.Logging.MetricStoreCapacity = 3600;
// defines the size of the event store in memory
backendConfiguration.Logging.EventStoreCapacity = 10000;
// defines the log level
backendConfiguration.Logging.LogEvents = LoggingLevel.Normal;
return backendConfiguration;
```

Offline Profiling

}

In offline profiling, all OpenAccess metrics and events are written to a log file, which can be reviewed in the Profiling and Tuning Wizard at a later time.

```
private static BackendConfiguration GetConfiguration()
    {
```



```
var backendConfiguration = new BackendConfiguration()
            {
                Backend = "mssql"
            };
            // defines the log level
            backendConfiguration.Logging.LogEvents =
LoggingLevel.Normal;
            backend.Logging.Downloader.Filename =
"c:\\QuickStartProfilingSession";
            backend.Logging.Downloader.EventBinary = true;
            backend.Logging.Downloader.MetricBinary = true;
            return backendConfiguration;
        }
```

```
Real-time & Offline Profiling
```

This will allow OpenAccess to report events and metrics to the profiler in real time, and it will also output all data to a log file, which can be reviewed at a later time.

```
private static BackendConfiguration GetConfiguration()
        {
            var backendConfiguration = new BackendConfiguration()
            {
                Backend = "mssql"
            };
            // defines the log level
            backendConfiguration.Logging.LogEvents =
LoggingLevel.Normal;
// defines the size of the metric store in memory
            backendConfiguration.Logging.MetricStoreCapacity = 3600;
            // defines the size of the event store in memory
            backendConfiguration. Logging.EventStoreCapacity = 10000;
            backend.Logging.Downloader.Filename =
"c:\\QuickStartProfilingSession";
            backend.Logging.Downloader.EventBinary = true;
            backend.Logging.Downloader.MetricBinary = true;
            return backendConfiguration;
        }
Note
```

Real-time profiling, and offline profiling can be used together, or separately.

5. Update the constructor to call the new GetConfiguration method created in step 4, rather than passing in the previously defined backendConfiguration object. As show here: public QuickStartEntitiesContext()

```
: base(DbConnection, GetConfiguration(), metadataSource)
```



6. The fluent data model is now configured for profiling.

Next Steps...

With the fluent model configured, the next step is to <u>connect the profiler</u> for real time profiling, or <u>view the logs generated</u> during offline profiling.

Configuring a Visual Designer Model (Forward or Reverse Mapping) for Profiling

To configure the profiler for online capture:

- 1. Create a data model using the Visual Designer section of this guide.
- 2. Add a reference to **Telerik.OpenAccess.ServiceHost.dll** in the project that will consume the domain model.
- 3. Open the OpenAccess Visual Designer.
- 4. Right click on the design surface, and select **Show Model Settings.**
- 5. In the model settings window navigate to **Backend Configuration > Tracing and**

			٠			
LO	g	g		n	g	•

📒 Model Settings		
Backend Configuratic	n Settings	ORM
Backend and Model Naming Rules Code Generation Backend Configuration Update Schema	Runtime Tracing & Logging Second Level Cache Conne Image: Comparison of the second device of the second devic	ection Pool ile E Keep Events for Profiler File Keep Metrics for Profiler
	File Settings Always Append New Sessions File Name Max File Size (KB) 1000 🗣 Events Threshold Metrics Threshold 0 🗣	Max Number of Backup Files 3 Events Store Interval 1 Metrics Store Interval 60
	Metrics Snapshot (msec)	OK Cancel

 Ensure Enable Logging is checked, and adjust the log level. By default only errors are reported by OpenAccess, to send more detail to the profiler, set the Log Level to Normal, Verbose, or All. Also, check Include Stack Trace so it is reported to the profiler.



Backend and Model	
Naming Rules	Runtime Tracing & Logging Second Level Cache Connection Pool
Code Generation	
Update Schema	Enable Logging
	Logging
	Log Level Verbose 🔻 🗹 Include Stack Trace
	Output Options
	Log Log Log Metrics to Binary File Keep Metrics for Profiler
	I Log Fuents to Text File
	File Settings
	File Name Events Store Interval 1
	Max File Size (KB) 1000 🔦 Metrics Store Interval 60 🔍
	Memory Buffer
	Events Threshold 1000
	Metrics Threshold 0
	Metrics Snapshot (msec) 0

7. Now the profiling method must be configured. OpenAccess provides the follow profiling methods:

Real-time Profiling

In real-time profiling, OpenAccess reports its events, and metrics, directly to the profiler via a service.

To enable real-time profiling, check **Keep Events for Profiler**, and **Keep Metrics for Profiler**.



Backend and Model	
Vaming Rules	Runtime Tracing & Logging Second Level Cache Connection Pool
Code Generation	
ackend Configuration	Enable Logging
puate schema	Logging
	Log Level Verbose 🔻 🖳 Include Stack Trace
	Coutput Options
	□ Log to Console □ Log Events to Binary File ☑ Keep Events for Profiler
	✓ Log to Trace □ Log Metrics to Binary File ✓ Keep Metrics for Profiler
	Log Events to Text File
	⊂ File Settinas
	Always Append New Sessions Max Number of Backup Files 3
	File Name Events Store Interval 1
	Max File Size (KB) 1000 🔪 Metrics Store Interval 60 🗬
	Events Threshold 10000
	Metrics Threshold 3600
	Metrics Snapshot (msec)

Offline Profiling

In offline profiling, all OpenAccess metrics and events are written to a log file, which can be loaded and reviewed in the Profiler and Tuning Advisor at a later time.

To enable offline mode, check **Log Events to Binary File** and **Log Metrics to Binary File**. This will enable the **File Settings** section. In the File name section, put the following path:

C:\QuickStartProfilingSession

OpenAccess will now log all events, and metrics a file at this location.

Note

The OpenAccess logger provides the ability to configure dynamic log file names, to learn more please read this <u>documentation article</u>.



Backend and Model	
Naming Rules Code Generation	Runtime Iracing & Logging Second Level Cache Connection Pool
Backend Configuration	Finable Logging
Update Schema	
	Log Level Verbose Vinclude Stack Trace
	Utput Options
	✓ Log to Trace ✓ Log Metrics to Binary File ✓ Keep Metrics for Profiler
	Log Events to Text File
	- Eila Sattingr
	Always Append New Sessions Max Number of Backup Files
	File Name c:\\QuickStartProfilingS Events Store Interval 1
	Max File Size (KB) 1000 🔪 Metrics Store Interval 60 🔍
	C Memory Buffer
	Events Threshold 10000
	Metrics Threshold 3600
	Metrics Snapshot (msec) 1000

Real-time & Offline Profiling

This will allow OpenAccess to report events and metrics to the profiler in real time, and it will also output all data to a log file, which can be reviewed at a later time.

To enable both modes, simply make the selections described in both of the above configurations.



Backend and Model Naming Rules Code Generation Update Schema Runtime Tracing & Logging Logging Log Level Verbose V Include Stack Trace Output Options Log to Console V Log Events to Binary File Log to Console Log to Trace Log to Trace Log Events to Text File File Settings File Stings File Name Keep Revices for Profiler Max File Size (KB) Metrics Store Interval Metrics Metrics Store Interval Metrics Metrics Store Interval Metrics Metrics Metrics Metrics M		
Logging Log Level Verbose Include Stack Trace Output Options Log to Console Include Stack Trace Log to Console Include Stack Trace Log to trace Include Stack Trace Log Events to Binary File Keep Events for Profiler Log Events to Text File File Settings File Settings File Name ickStartProfilingSession Max Number of Backup Files 3 * File Name ickStartProfilingSession Max File Size (KB) 1000 * Metrics Store Interval 60 * Metrics Threshold 1000 * Metrics Threshold 1000 *	Backend and Model Naming Rules Code Generation Backend Configuration Update Schema	Runtime Tracing & Logging Second Level Cache Connection Pool Image: Tracing & Logging Image: Tracing & Logging Image: Tracing & Logging
File Settings 3 Always Append New Sessions Max Number of Backup Files File Name ickStartProfilingSession Max File Size (KB) 1000 Memory Buffer Events Threshold Metrics Threshold Metrics Snapshot (msec)		Log Level Verbose Ug Log Log Log Log Log Log Log Log Log Events to Binary File
Memory Buffer Events Threshold Metrics Threshold 3600 Metrics Snapshot (msec)		File Settings Image: Setting set
		Memory Buffer Events Threshold Metrics Threshold 3600 Metrics Snapshot (msec)

Note

Real-time profiling, and offline profiling can be used together, or separately.

8. Now click **OK**. At this point the model is configured to report SQL Events, and Metrics.

Next Steps...

With the fluent model configured, the next step is to <u>connect the profiler</u> for real time profiling, or <u>view the logs generated</u> during offline profiling.

Real-time Profiling

Applications using OpenAccess ORM can report SQL events, and metrics to the Profiler and Tuning Advisor in real time. To do this, first a service must be added to the application consuming the OpenAccess Data Model. Then the profiler must be configured to connect to the service.

Configuring an Application

1. When the application is starting, the profiler service needs to started.

In a web application, this should be pasted in Application_Start

In a **windows application**, paste this **before** showing the first form.



Telerik.OpenAccess.ServiceHost.ServiceHostManager.StartProfilerService(15555);

This will start the profiler service on port 15555. A different port can be used by passing a different port number to the StartProfilerService method.

2. When the application is ending, the service needs to be stopped by calling:

Telerik.OpenAccess.ServiceHost.ServiceHostManager.StopProfilerService()

In a web application, this should be added in Application_End

In a **windows application**, this should be added to the closing event for the main form.

3. **Run** the application.

Connecting the Profiler

- 1. Make sure the application is running.
- 2. **Open** the profiler, by navigating to **Telerik > OpenAccess > Profiler and Tuning** Advisor in Visual Studio



3. Once the profiler opens, **click** the **Connect To** menu item



4. Configure the connection



- a. Server The server where your application is running. In most cases this will be localhost.
- b. Port The service host added to the application earlier uses a specific port to communicate with the profiler. This port is what is passed into StartProfilerService(*port*);
- c. Connection Name– The profiler needs to know which connection to monitor. Specifying the connection name gives the profiler the needed information to properly report activity.

Open Connection	X
Server:	localhost
Port:	15555
Connection name:	QuickStartConnection 🔹
Test Connection	Connect Cancel

5. Once configured, click **Test Connection** at which point the profiler will indicate whether or not it was able to successfully connect to the service host.

If **Test Connection** reports a **failure**, ensure that the application is running, and that you specified correct info on the **Open Connection** dialog.

- 6. Once **Test Connection** comes back successful, click **Connect**.
- 7. At this point the application being profiled can be used, and all activity will be reported to the profiler.





For more information on all the available features of the profiler make sure to check out this <u>documentation article</u>.

Viewing Offline Profiling Session Logs

 Open OpenAccess Profiler and Tuning Advisor from the start menu by navigating to Start>All Programs >Telerik > OpenAccess ORM > Profiler and Tuning Advisor.



2. Click the **Folder** icon in the toolbar of the profiler.



OpenAccess Profiler	
	Switch views 📑 🗸
No Data Series	
0:00	00:00
SQL Events LINQ/API Events Alerts Grouped SQL Events	Details Stack Trace
Drag a column header and drop it here to group by that column	SQL Statement:
SQL Start Time Duration (ms) Rows Context Id Context Name	
	Parameter name: Value:
vents loaded: 0 Metrics loaded: 0	Status: Ready

- 3. **Navigate** to the directory where the log was saved, for example, in this guide it was saved to the root C drive.
- 4. Select the .oalog file.

The OpenAccess logger produces two files:

- a. .oalog contains all of the operation information.
- b. .oametrics contains all of the counters for the OpenAccess context.
 Things like context lifetime, inserts per second, selects per second, cache hits, cache misses, etc. are stored in this file.
- The log should now be ready for to explore in the OpenAccess Profiler and Tuning Advisor

For more information on all the available features of the profiler make sure to check out this <u>documentation article</u>.

