Contents

This sample shows a simple implementation of the DevForce *IDataSourceKeyResolver* interface. This interface allows your application to take control of the *key resolution* process, allowing you to programmatically determine the *DataSourceKey* used to connect to the run time database.

- **Platform:** Console
- **Language:** C#, VB
- **Download:** Custom DataSourceKeyResolver (Console)

Problem

Many applications may want to omit connection information from configuration files and programmatically supply dynamic connections strings at run time.

Solution

By implementing a custom *IDataSourceKeyResolver* you can take control of key resolution in your application.

Running the sample

A simple console application, the application first constructs an *EntityManager* for the "Tomato" data source name and the "Customer01" data source extension.

```
var entityManager = new DomainModelEntityManager(true, "Customer01");
```

```
Dim entityManager = New DomainModelEntityManager(True, "Customer01")
```

When the query is executed, DevForce, having found our *IDataSourceKeyResolver*, will call its *GetKey* method to resolve the key name and extension into a *DataSourceKey*. Looking at the CustomDataSourceKeyResolver, we see that it queries the Tenant database with the key name and extension provided to find the database connection string to use. With the connection string in hand it then builds a *ClientEdmKey*, a type of *DataSourceKey* representing an Entity Data Model, and returns that key to DevForce. DevForce then uses the tenant-specific connection information whenever queries or saves are performed against the "Tomato" data source for "Customer01".

After querying for "Customer01" the sample then goes through the same steps for "Customer02":

```
var entityManager2 = new DomainModelEntityManager(true, "Customer02");
```

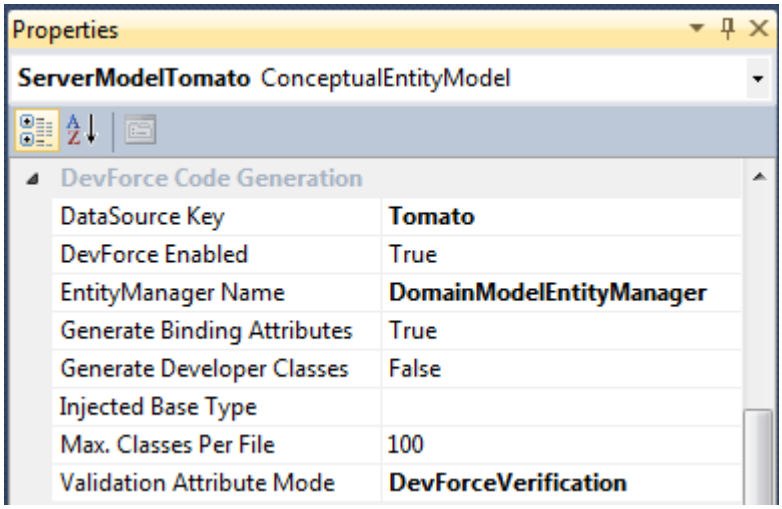
```
Dim entityManager2 = New DomainModelEntityManager(True, "Customer02")
```

For additional insight into the workings of the *CustomDataSourceKeyResolver*, set a breakpoint in its *GetKey* method to step through its logic as it's called by DevForce.

Configuration

Note there are two "model" projects within this solution: ServerModelTenant and ServerModelTomato. Ignoring the somewhat wordy names, the first is a "raw" Entity Framework model for the Tenant database, while the second is a DevForce model for the Tomato database. Why is the Tenant model raw EF? We'll be accessing the Tenant database for customer-specific connection information in this sample and it's a bit easier not to use DevForce to connect to this model, plus it shows that it's easy to have both EF and DevForce models in a solution.

Next let's look at the DevForce-specific properties on the ServerModelTomato model, specifically noting that the "DataSource Key" is called "Tomato".



DevForce writes this information into the auto-generated model. If you open the *.IB.Designer file you'll see the DataSourceKeyName attribute on generated classes:

```
[IbEm.DataSourceKeyName(@"Tomato")]
public partial class DomainModelEntityManager : IbEm.EntityManager
```

```
<IbEm.DataSourceKeyName("Tomato")>
Partial Public Class DomainModelEntityManager
 Inherits IbEm.EntityManager
```

By default, at run time DevForce will take this key name, and combined with the *data source extension* provided to the *EntityManager*, build a run time data source key. Here, because we've supplied a custom *IDataSourceKeyResolver*, DevForce delegates resolution of this key to our resolver. The sample resolver here does a simple query on the "Tenant" database to find the appropriate connection information for a given key name and extension.

The application will be querying the Tomato01 and Tomato02 databases, but instead of providing connection strings directly in the config file we'll dynamically build them here using the *CustomDataSourceKeyResolver*.

Note that only the "tenant" key includes a connection string in the config file.

Prerequisites

Attach the two databases in the zip file TomatoDbInstances.zip found in the Data folder. These are named Tomato01 and Tomato02, and contain a single table named Tomato. The databases are identical in schema but have some differences in data. They are meant to represent a database for which a separate copy is supplied for each of your customers.

Also attach the Tenant database contained in the zip file Tenant.zip found in the Data folder. This database contains one table, "DataSourceConnection" that maps a ConnectionString to a DataSourceKeyName and DataSourceExtension. The CustomDataSourceKeyResolver in this application will query this table to find connection information when needed.