Contents

- The need for different data source targets
- DataSourceKeyName
- DataSourceExtension
- Construct an EntityManager with a DataSourceExtension
- Resolving the DataSourceExtension on the server

You can **target different databases dynamically** by creating the *EntityManager* with a non-default *DataSourceExtension*.

The need for different data source targets

When you start writing an application, you probably only have one database, your development database.

That won't last long. Eventually, the application will query and save to one database during development, and to other databases in your staging, test, and production environments. That's four different database instances (all with the same schema one hopes). You need some way to switch among these databases depending upon which the environment you want to be in.

You may put the "switch" in a configuration file on the client. You might determine it programmatically on the client. Either way, the client application should be able acquire the value of the switch and choose the appropriate server environment.

Let's generalize this thought. Suppose that you are building a multi-tenant application with a different database for each tenant. There will be far more than four databases.

You probably don't want to set up different servers for each one. It is easier to have a single (load-balanced) server address and tell the server at that address which tenant database to use. User input can provide the client application with the information it needs to determine which tenant environment to use. Now we have to tell the server about it.

The EntityManager tells the server which data source to use by sending two types of information in each request:

1. The *DataSourceKeyName* that identifies the data source schema
2. The *DataSourceExtension* that identifies which one of the schema-matching databases to use.

DataSourceKeyName

Every entity model is associated with a data source. The data source is typically a database and we'll assume it is a database in this discussion. The schema of that database maps to the entity classes in the model. You could say that the schemas of the entity model and the database are matched.

However, the entity model doesn't know which concrete database holds the actual data for the application. Your application code shouldn't know either. The actual database to use at runtime is determined by a connection string; that's a configuration concern best relegated outside the client code base.

A distributed client application (e.g., a Silverlight application) should never contain database connections strings of any kind. The connection string is never needed on the client and it's a serious security violation to have one there.

Instead a DevForce application refers to the database by name, by its *DataSourceKeyName* resolution logic on the server determines which connection string to use based on the *DataSourceKeyName* associated with the entity classes in the request.

For example, the *DataSourceKeyName* could be "NorthwindEntities" as it often is in our sample code. We know that's the key because we typed that name when we created the entity model. We can also tell by inspecting the attributes on the generated EntityManager and entity class files:

```
IbEm.DataSourceKeyName(@"NorthwindEntities")]
public partial class NorthwindEntities : IbEm.EntityManager {}
[IbEm.DataSourceKeyName(@"NorthwindEntities")]
public partial class Customer : IbEm.Entity {}
```

```
<IbEm.DataSourceKeyName("NorthwindEntities")>
Partial Public Class NorthwindEntities
Inherits IbEm.EntityManager
End Class
<IbEm.DataSourceKeyName("NorthwindEntities")>
Partial Public Class Customer
Inherits IbEm.Entity
End Class
```

DataSourceExtension

The *DataSourceKeyName* identified the database schema to use - some kind of Northwind database - but it didn't tell us which concrete database to use. It doesn't tell us which of many possible versions of the Northwind database to use.

That's the job of the *DataSourceExtension*. The *DataSourceExtension* is a string that DevForce combines with the *DataSourceKeyName* to determine the concrete database to use. The "NorthwindEntities" *DataSourceKeyName* tells the server that the client wants some version of a Northwind database; the value of the *DataSourceExtension* tells the server which specific Northwind database.

The *DataSourceExtension* string value is up to you. You devise your own extension naming scheme. It might be "Dev", "Test", "Stage" and "Prod". It could be the Tenant ID.

The trick is in what you do with that string

Construct an EntityManager with a *DataSourceExtension*

To create an *EntityManager* using a datasource extension, simply pass the name of the datasource extension as a parameter in the constructor:

```
manager = new NorthwindEntities(dataSourceExtension: "Test" );
```

```
manager = New NorthwindEntities(dataSourceExtension: "Test" )
```

Henceforth, this *manager* is dedicated to the "Test" environment and will always tell the server to query and save to the "Test" version of the database.

Note that the **'+'** sign is a reserved character that cannot be used in a DataSourceExtension string. If you do, you will receive the following error: **IdeaBlade.Core.IdeaBladeException: Unable to find a compositionContext with the name: ....**.

Resolving the *DataSourceExtension* on the server

To resolve the connection string, the server will look in the .config file and look for a connection string name with the format "DataSourceKeyName_DataSourceExtensionName"

```
<connectionStrings>
   <add name="NorthwindEntities_Test"
       connectionString="…;Initial Catalog=NorthwindTest…" />
   <add name="NorthwindEntities_Production"
       connectionString="…;Initial Catalog=NorthwindProduction;…" />
</connectionStrings>
```

If it does not find this, then it will search for a "parent" connection string by removing the extension.

Data source extensions are hierarchical: you can separate each segment with an underscore ("_") character. For example, you could use an extension of "Test_A" to indicate tenant "A" in the "Test" environment. If a data source key for "Test_A" is not found, then DevForce will look for "Test", and if that is not found then the default data source key name is used.