Contents

Ready to migrate your DevForce Classic application to the latest generation of DevForce? This guide provides an overview of the areas that you may need to consider when planning your migration effort.

Data Schema Design

DevForce's object mapping is now directly integrated with the Entity Framework's designer. You will need to remap your model using the Entity Framework designer to create an Entity Data Model (EDM) and then DevForce will generate the business object classes for you. You then migrate your business logic by adding it to partial classes to complete the implementation. You can also take this opportunity to fine tune the Entity Data Model and cover up possible shortcomings in the original database schema.

To learn more about creating an Entity Data Model, see Start with the EDM Wizard.

Querying

Queries written using DevForce Classic's Object Query Language will need to be rewritten to use LINQ. This is a great opportunity to use LINQ's more expressive and compact syntax as well as gain some additional compile-time type checking. For more details, see Query using LINQ.

PassthruSql in DevForce Classic will need to be converted to PassthruESQL which will work against the Entity Data Model. This should translate cleanly, but you may need to make a few adjustments if you fine tuned the EDM mapping to the database.

Async

DevForce now supports the new async/await keywords which make asynchronous programming much easier by making it feel like you're writing synchronous code. So, instead of writing callbacks, you now use *await*. For example:

```
async void MainPage_Loaded(object sender, RoutedEventArgs e) {
  var mgr = new NorthwindIBEntities();
  var emps = await mgr.Employees.ExecuteAsync();
  ...
}
```

See Query asynchronously to learn more.

Concurrency Columns

DevForce now supports a richer set of concurrency columns than were available in DevForce Classic. The available choices are: None, AutoGuid, AutoDateTime, AutoIncrement, ServerCallback, and Client.

Sql Where Clauses

This DevForce Classic capability is now handled by the Entity Framework, in the SSDL or storage schema model.

User Defined Columns

User defined columns are not supported by the Entity Framework. One strategy to handle this is to convert the custom columns into rows on a table that represents extension properties of the object.

Dynamic Entities

Dynamic Entities in their original DevForce Classic form cannot be supported by the Entity Framework. However, DevForce has now augmented the Entity Framework so that you can perform arbitrary projections into known or anonymous types and still cache and bind to them.

Persistence Order

Persistence order in DevForce is now handled by the underlying Entity Framework.

UI

If you have been following the principles of composability and modularity, your UI should be largely unaffected. The WinForms databinding APIs are identical in the new version of DevForce. However, you may wish to explore the possibility of enhancing the user experience with WPF. WPF is very powerful, but also has a steep learning curve. If you are interested in WPF, we have some experts that may be able to help get you started.

Note that the **WinForms databinders** are now installed as a separate package that you can find on the Download Portal.

Installation

Like Microsoft's Entity Framework, DevForce has made the leap to NuGet. Instead of running a large installer, you now use NuGet to add a DevForce package to your project. NuGet will resolve all dependencies and automatically adds all the libraries and references that you'll need. For more details, see Installing DevForce.

DevForce and DevForce Classic can be installed at the same time. You shouldn’t have any issues using them both on the same machine.

Need more help?

Our professional services team can help customers migrating from DevForce Classic. Contact us to learn more.