#### **Contents**

- Problem
- Solution
  - Attribute interception
  - Dynamic property interception
  - Run the app

You can easily extend the behavior of entities in your model using property interceptors. We'll show you how here.

• Platform: Silverlight, WPF

• Language: C#, VB

• Download: Property interceptors (Silverlight), Property interceptors (WPF)

## **Problem**

You'll often want to customize your model with property interceptors to extend the behavior of your entities.

### Solution

This simple application shows how to declare and work with property interceptors in a Silverlight application. In this sample the two primary ways of creating property interceptors are shown.

# Attribute interception

In the **Customer.cs** file, you'll find several interception methods, all marked with a property interceptor attribute. We've only provided a few methods, but this should give you an idea of what you can do:

- · UppercaseCountry marked with AfterGet for the Country property. Note the method signature used here.
- AfterSetAnyProperty marked with AfterSet for any property in the Customer class. We just provide simple logging
  here
- BeforeSetCompanyName marked with BeforeSet for the CompanyName property. This sample forces the incoming value to upper case.
- AfterSetCompanyName marked with AfterSet for the CompanyName property. Provides simple logging.

#### **Dynamic property interception**

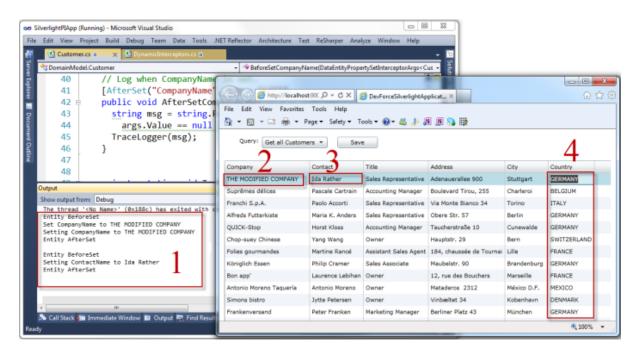
The **DynamicInterceptors.cs** file contains several examples showing how to dynamically add interceptors at run time. Interceptors can also be removed and temporarily skipped at run time too, but we don't show that here.

Two actions are added here, both on Entity. Interception actions on a type apply to that type and all sub-types. Here we provide simple logging on a *BeforeSet* and *AfterSet* for any property on any entity.

Unlike the attribute interceptors in the Customer type, which DevForce discovers automatically, we need to specifically add these dynamic interceptors at run time. The AddDynamicEntityInterceptors method is called in Page.xaml.cs when initializing persistence activities.

### Run the app

When you run the application in the Visual Studio debugger (e.g., with F5), you see the results of the interceptors in action.



- 1. The Visual Studio Output Window displays messages written by the interceptors in both *Customer.cs* and in *DynamicInterceptors.cs*
- 2. When the user changed the company name to "The Modified Company", the CompanyName "set" interceptor in *Customer.cs* converted that value to upper case.
- 3. The user changed the Contact to "Ida Rather". There are no interceptors for that property specifically but the *Entity* level property interceptor in *DynamicInterceptors.cs* logged the change to the Output window (1).
- 4. All "Country" values are displayed in upper case because of the Country "get" interceptor in Customer.cs