You can also **define an interceptor to work across more than one type**; e.g., against multiple entity types. In this event, you will want to locate the interceptor in a separate (non-entity) class; and you will need to tell DevForce explicitly to discover the interceptors therein in order to activate them.

Here's an AfterGet interceptor that intercepts every Get against every property of the Employee and Product entities. (Don't overuse this technique, or you may slow down your app unacceptably!) The interceptor checks to see if the property is a DateTime, or something derived from a DateTime, and that its value is not null. If both things are true, it converts the DateTime to local time. (The interceptor assumes that the DateTime has been stored in the database as a universal time.)

```
[IbCore.AfterGet(TargetType = typeof(Employee))]
[IbCore.AfterGet(TargetType = typeof(Product))]
public static void UniversalGetInterceptor(IbEm.IEntityPropertyInterceptorArgs args) {
 bool isDateTime = args.EntityProperty.DataType.IsAssignableFrom(typeof(DateTime));
 bool valueIsNull = args.Value == null;
 if (isDateTime && !valueIsNull) {
    DateTime aDateTime = ((DateTime)args.Value);
    aDateTime = aDateTime.ToLocalTime();
    args.Value = aDateTime;
  }
}
```

```
<IbCore.AfterGet(TargetType:=GetType(Employee)), _
IbCore.AfterGet(TargetType:=GetType(Product))> _
Public Shared Sub UniversalGetInterceptor(ByVal args As IbEm.IEntityPropertyInterceptorArgs)
 Dim isDateTime As Boolean = args.EntityProperty.DataType.IsAssignableFrom (GetType(Date))
 Dim valueIsNull As Boolean = args.Value Is Nothing
 If isDateTime AndAlso (Not valueIsNull) Then
   Dim aDateTime As Date = (CDate(args.Value))
    aDateTime = aDateTime.ToLocalTime()
    args.Value = aDateTime
 End If
End Sub
```

Note that, since we defined the above interceptor in a non-entity class, we had to specify in the AfterGet attributes the types against which it is to operate: here, Employee and Product.

DevForce won't find interceptors defined outside entity classes automatically. In these cases you need to tell the *PropertyInterceptorManager* where to look. Here's how you let DevForce know you've defined some interceptors in a class named *AuxiliaryPropertyInterceptors*:

```
IbCore.PropertyInterceptorManager.CurrentInstance.DiscoverInterceptorsFromAttributes(typeof(AuxiliaryPropertyInterceptors));
```

```
IbCore.PropertyInterceptorManager.CurrentInstance.DiscoverInterceptorsFromAttributes(typeof(AuxiliaryPropertyInterceptors))
```