Contents

- Standard interceptor action
- Alternative interceptor action

While the property interceptor methods described previously allow a great deal of control over the entire interception process, there are times when this is overkill. Sometimes all you really want is to do is modify or inspect the incoming or outgoing values. In these cases, a simplified signature for an interception method is also provided.

Standard interceptor action

Code Listing 14. Employee.UppercaseLastName().

```
[IbCore.AfterGet(Employee.EntityPropertyNames.LastName)]
public void UppercaseLastName(
  IbCore.PropertyInterceptorArgs<Employee, String> args) {
  var lastName = args.Value;
 if (!String.IsNullOrEmpty(lastName)) {
    args.Value = args.Value.ToUpper();
  }
}
```

```
<IbCore.AfterGet(Employee.EntityPropertyNames.LastName)>
Public Sub UppercaseLastName(ByVal args As  _
  IbCore.PropertyInterceptorArgs(Of Employee, String))
 Dim lastName = args.Value
 If Not String.IsNullOrEmpty(lastName) Then
    args.Value = args.Value.ToUpper()
 End If
End Sub
```

Alternative interceptor action

Code Listing 15. Employee.UppercaseLastNameV2().

```
[IbCore.AfterGet(Employee.EntityPropertyNames.LastName)]
public String UppercaseLastNameV2(String lastName) {
 if (!String.IsNullOrEmpty(lastName)) {
   return lastName.ToUpper();
  } else {
   return String.Empty;
  }
}
```

```
<IbCore.AfterGet(Employee.EntityPropertyNames.LastName)>
Public Function UppercaseLastNameV2(ByVal lastName As String) As String
 If Not String.IsNullOrEmpty(lastName) Then
   Return lastName.ToUpper()
 Else
   Return String.Empty
 End If
End Function
```

In general, any property interceptor action that only inspects or modifies the incoming value without the need for any other context can be written in this form. In fact, if the action does not actually modify the incoming value, the return type of the interceptor action can be declared as void.