PropertyAccessor.cs 屬性存取器

靈活的操作資料物件-複製,讀取及寫入

Created by Andy.flower 2021/10/2

PropertyAccessor.cs

- 主要功能:
  可以透過"PropertyName" 存取資料物件的值

- 語法:
  var accessor = PropertyAccessor.Create(Customer)
  accessor["CustomerID"] = value;
  object customerid = accessor["CustomerID"];
  string customerid = accessor.Get<string>("CustomerID");

方法列表

- this[string propertyName]
- Get<T>(string propertyName)
- GetValues(params string[] propertyNames)
- GetValues()
- GetKeyValues(params string[] propertyNames)
- GetKeyValues()
- SetValues(IDictionary<string, object> keyvalues)
- SetValues(string[] propertyNames, object[] values)
- SetValues(object[] values)
- ContainsKey(string propertyName)
- SetNullValuesToDefault()

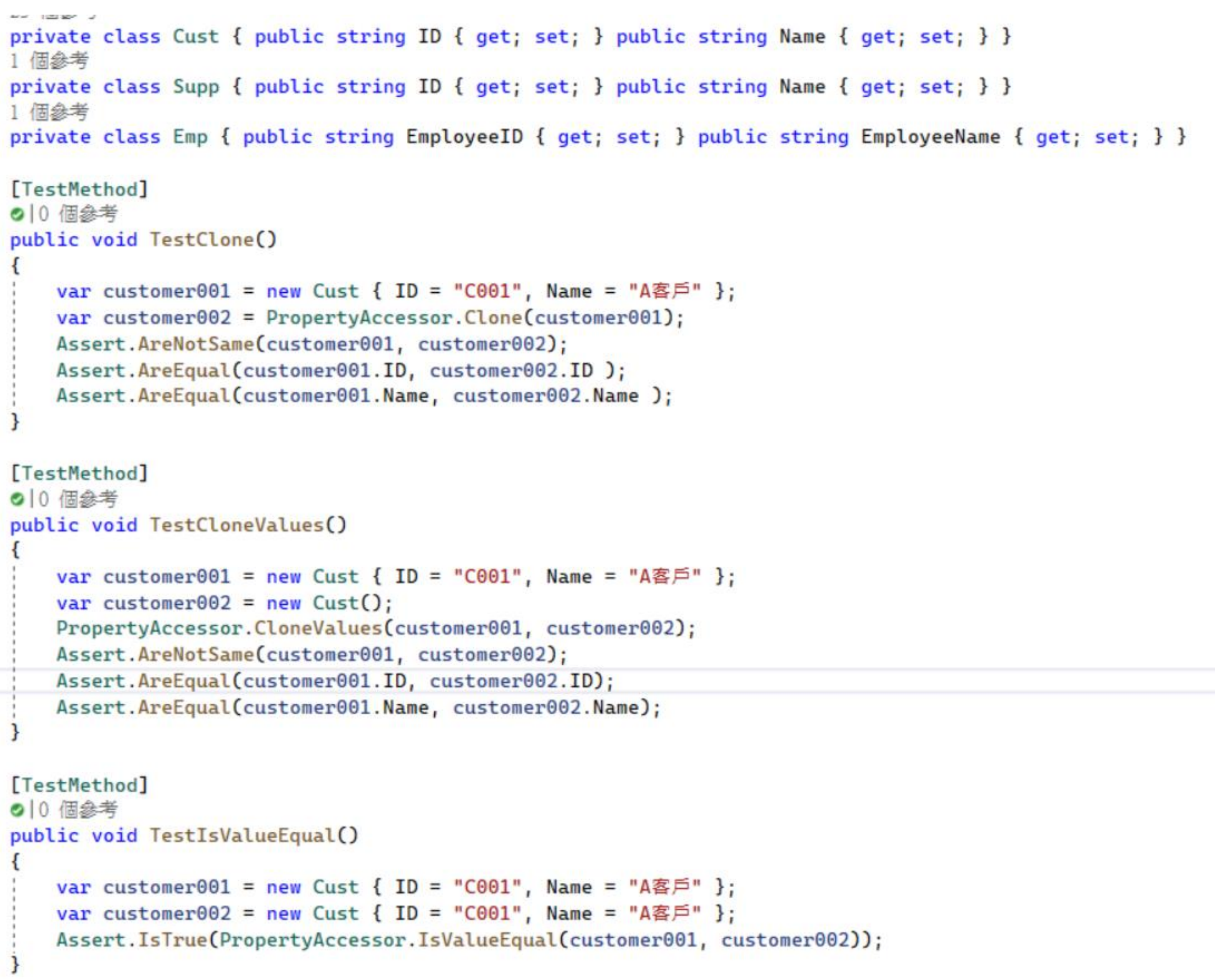
提供的靜態方法(1/2)

- Create(object)
  利用object建立一個屬性存取器
- Create<T>()
  利用 class 建立一個屬性存取器
- Clone<T>(T source)
  利用現有的物件, 複製出一個新的物件
- CloneValues<T>(T source, T destination)
  將物件的屬性值複製到另一個的物件
- CloneMany<T>(IEnumerable<T> sources)
  利用現有的 IEnumerable, 回傳一個新 Ienumerable
- IsValueEqual(object self, object other)
  檢查兩個相同型別物件的所有屬性值是否相同

提供的靜態方法(2/2)

- IsManyValueEqual<T>(IEnumerable<T> selfs, IEnumerable<T> others)
  檢查兩個集合物件的所有元素的屬性值是否相同
- AutoMapTo(object source, object dest, params string[] ignoreProps)
  將不同型別物件, 相同Property 的值由 source to destination
- MapTo(object source, string[] sourceProps, object dest, string[] destProps)
  不同型別物件, 自行定義要複製的 Property
- ContainsKey(string propertyName)
  檢查屬性名稱是否存在 - propertyName 不分大小寫
- Comparison<T>(T data1,T data2, string prop)
  針對兩個物件的屬性值比較大小

範例暨測試(1/4)



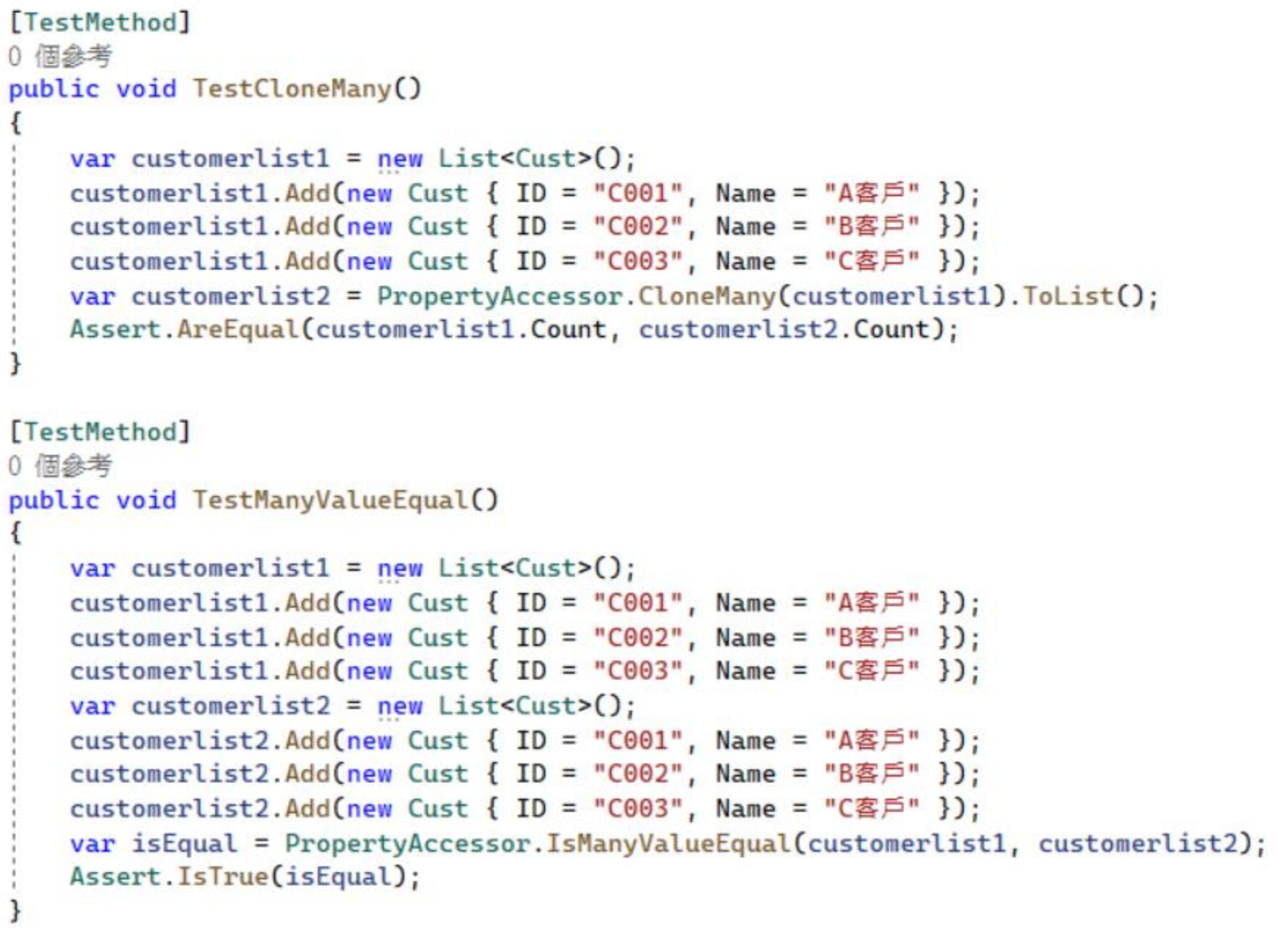
```
private class Cust { public string ID { get; set; } public string Name { get; set; } }
1 個參考
private class Supp { public string ID { get; set; } public string Name { get; set; } }
1 個參考
private class Emp { public string EmployeeID { get; set; } public string EmployeeName { get; set; } }

[TestMethod]
|0 個參考
public void TestClone()
{
    var customer001 = new Cust { ID = "C001", Name = "A客戶" };
    var customer002 = PropertyAccessor.Clone(customer001);
    Assert.AreNotSame(customer001, customer002);
    Assert.AreEqual(customer001.ID, customer002.ID );
    Assert.AreEqual(customer001.Name, customer002.Name );
}

[TestMethod]
|0 個參考
public void TestCloneValues()
{
    var customer001 = new Cust { ID = "C001", Name = "A客戶" };
    var customer002 = new Cust();
    PropertyAccessor.CloneValues(customer001, customer002);
    Assert.AreNotSame(customer001, customer002);
    Assert.AreEqual(customer001.ID, customer002.ID);
    Assert.AreEqual(customer001.Name, customer002.Name);
}

[TestMethod]
|0 個參考
public void TestIsValueEqual()
{
    var customer001 = new Cust { ID = "C001", Name = "A客戶" };
    var customer002 = new Cust { ID = "C001", Name = "A客戶" };
    Assert.IsTrue(PropertyAccessor.IsValueEqual(customer001, customer002));
}
```

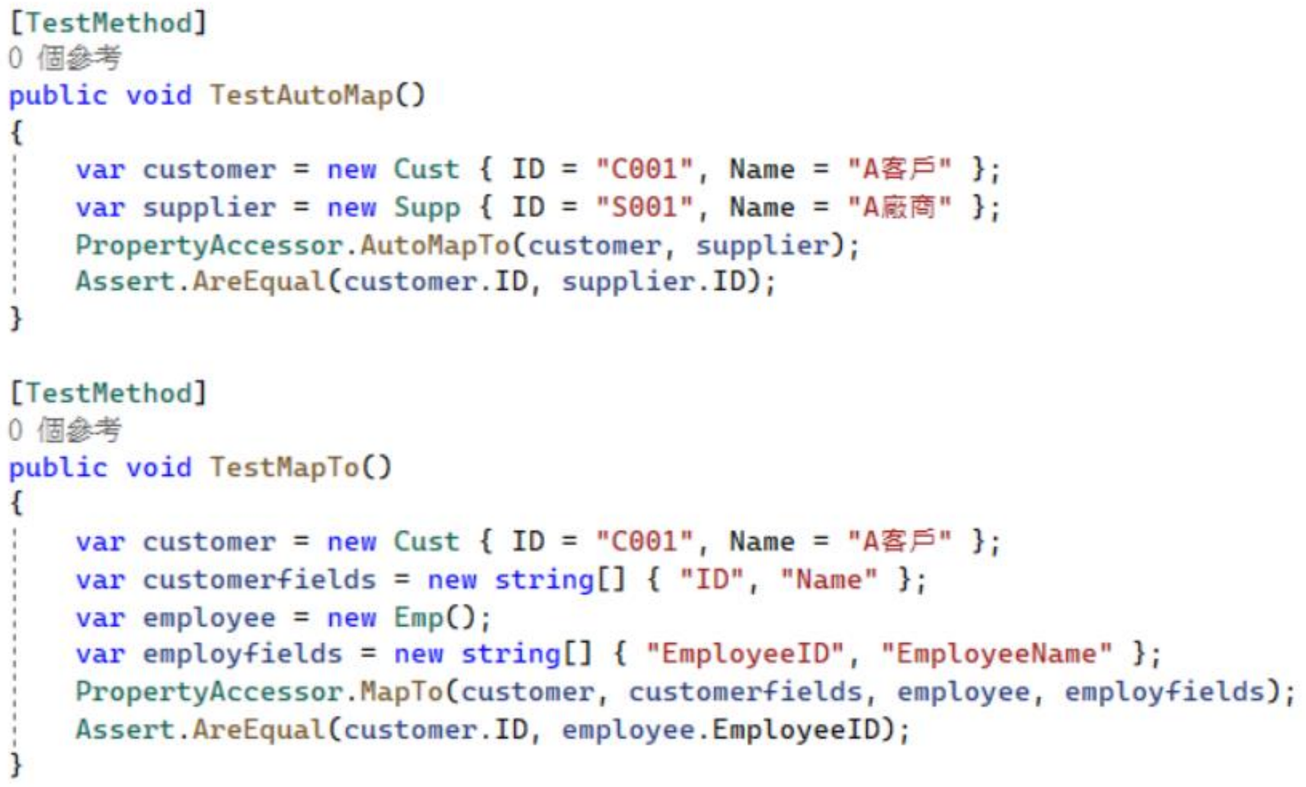
範例暨測試(2/4)



```
[TestMethod]
0 個參考
public void TestCloneMany()
{
    var customerlist1 = new List<Cust>();
    customerlist1.Add(new Cust { ID = "C001", Name = "A客戶" });
    customerlist1.Add(new Cust { ID = "C002", Name = "B客戶" });
    customerlist1.Add(new Cust { ID = "C003", Name = "C客戶" });
    var customerlist2 = PropertyAccessor.CloneMany(customerlist1).ToList();
    Assert.AreEqual(customerlist1.Count, customerlist2.Count);
}

[TestMethod]
0 個參考
public void TestManyValueEqual()
{
    var customerlist1 = new List<Cust>();
    customerlist1.Add(new Cust { ID = "C001", Name = "A客戶" });
    customerlist1.Add(new Cust { ID = "C002", Name = "B客戶" });
    customerlist1.Add(new Cust { ID = "C003", Name = "C客戶" });
    var customerlist2 = new List<Cust>();
    customerlist2.Add(new Cust { ID = "C001", Name = "A客戶" });
    customerlist2.Add(new Cust { ID = "C002", Name = "B客戶" });
    customerlist2.Add(new Cust { ID = "C003", Name = "C客戶" });
    var isEqual = PropertyAccessor.IsManyValueEqual(customerlist1, customerlist2);
    Assert.IsTrue(isEqual);
}
```

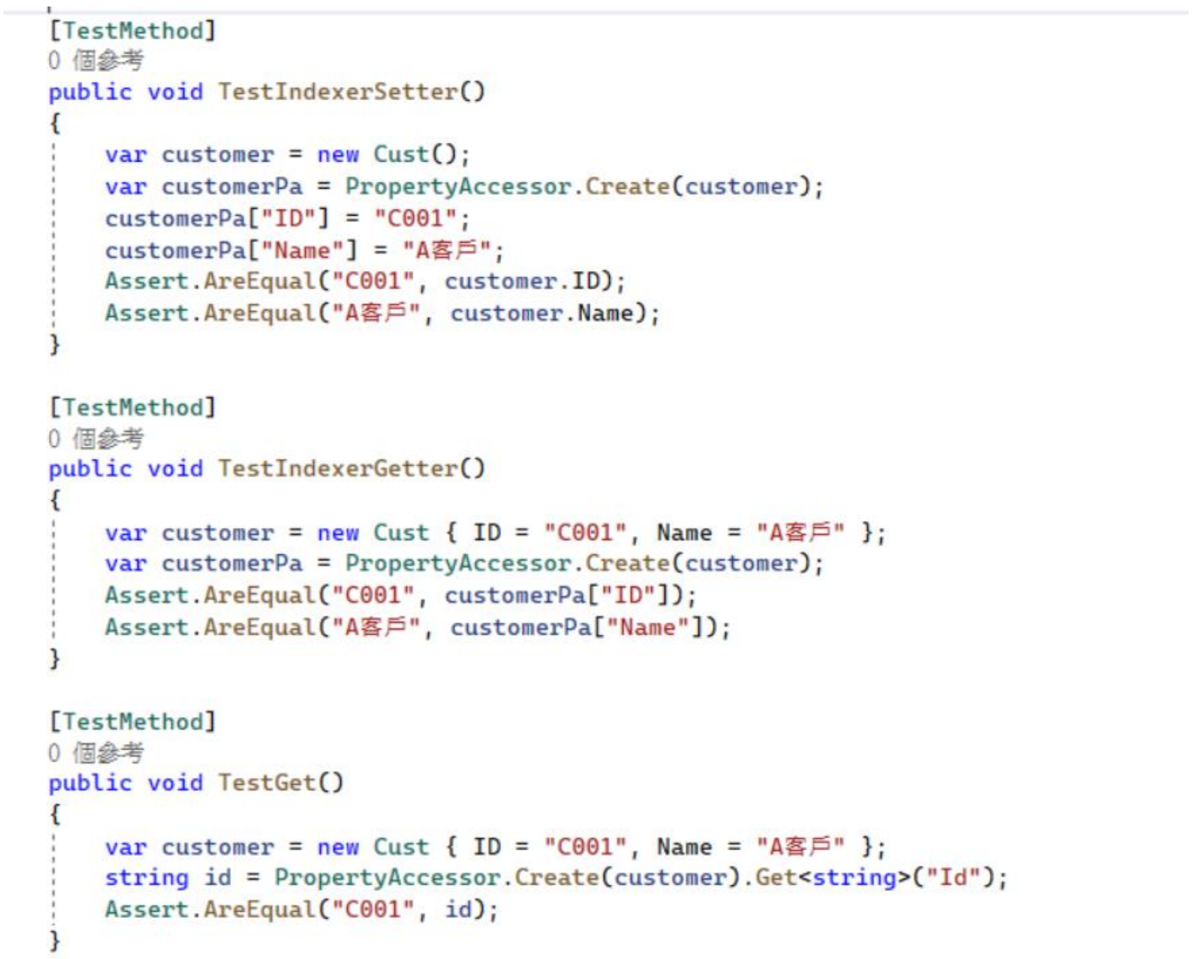
範例暨測試(3/4)



```
[TestMethod]
0 個參考
public void TestAutoMap()
{
    var customer = new Cust { ID = "C001", Name = "A客戶" };
    var supplier = new Supp { ID = "S001", Name = "A廠商" };
    PropertyAccessor.AutoMapTo(customer, supplier);
    Assert.AreEqual(customer.ID, supplier.ID);
}

[TestMethod]
0 個參考
public void TestMapTo()
{
    var customer = new Cust { ID = "C001", Name = "A客戶" };
    var customerfields = new string[] { "ID", "Name" };
    var employee = new Emp();
    var employfields = new string[] { "EmployeeID", "EmployeeName" };
    PropertyAccessor.MapTo(customer, customerfields, employee, employfields);
    Assert.AreEqual(customer.ID, employee.EmployeeID);
}
```

範例暨測試(4/4)



```
[TestMethod]
0 個參考
public void TestIndexerSetter()
{
    var customer = new Cust();
    var customerPa = PropertyAccessor.Create(customer);
    customerPa["ID"] = "C001";
    customerPa["Name"] = "A客戶";
    Assert.AreEqual("C001", customer.ID);
    Assert.AreEqual("A客戶", customer.Name);
}

[TestMethod]
0 個參考
public void TestIndexerGetter()
{
    var customer = new Cust { ID = "C001", Name = "A客戶" };
    var customerPa = PropertyAccessor.Create(customer);
    Assert.AreEqual("C001", customerPa["ID"]);
    Assert.AreEqual("A客戶", customerPa["Name"]);
}

[TestMethod]
0 個參考
public void TestGet()
{
    var customer = new Cust { ID = "C001", Name = "A客戶" };
    string id = PropertyAccessor.Create(customer).Get<string>("Id");
    Assert.AreEqual("C001", id);
}
```