**Story**

|  |
| --- |
| DSL |
| begin story “Title”  begin setup  **Interfaces** ...  Can be used in isolation  end setup  **Tests** ...  end story |
| C# |
| #region Using Statements  using System.Collections.Generic;  using System.Diagnostics;  using bdUnit.Interfaces;  using NUnit.Framework;  using Rhino.Mocks;  using StructureMap;  using System;  #endregion  namespace bdUnit.Interfaces  {  **Interfaces** ...  }  namespace bdUnit.Tests  {  [TestFixture]  public class Title  {  [TestFixtureSetUp]  public void Setup()  {  StructureMapInitializer.Initialize();  }    **Tests** ...  }  } |

**Interface**

|  |  |
| --- | --- |
| DSL | C# |
| @**User** to have a ~**Spouse**(@**User**)   and a ~**Name**  and a ~**Married**(false)  and an ~**IsDead**(false)  and an ~**IsActive**(false)  and an ~**Age**(0)  and a ~**CreatedDate**("01/01/1700")  and a ~**Location**(@**Location**)  and several ~**Children**(@**User**)  I want a @**User** to be able to #**Kill** another @**User** I want a @**User** to be able to #**ProposeTo** another @**User** I want a @**User** to be able to #**Marry** another @**User**  I want a @**User** to be able to #**Meet** several @**User** I want to be able to #**Save** @**User** | public partial interface IUser  {  IUser Spouse {get; set;}  string Name {get; set;}  bool Married {get; set;}  bool IsDead {get; set;}  bool IsActive {get; set;}  int Age {get; set;}  DateTime CreatedDate {get; set;}  ILocation Location {get; set;}  IList<IUser> Children {get; set;}  void Kill(IUser user);  void ProposeTo(IUser user);  void Marry(IUser user);  void Meet(IList<IUser> user);  void Save();  } |

**Test**

|  |
| --- |
| DSL |
| When a @**User**(Peter) #**ProposeTo** another @**User**(Patty)  and @**User**(Peter) #**Marry** another @**User**(Patty)  and @**User**(Peter) ~**Name** is ("Peter")  and @**User**(Patty) ~**Name** is ("Patty"),  each @**User** should have ~**Spouse** as the other @**User**  and should have ~**Married** as (true)  and should have ~**Age** less than (21) |
| C# |
| [Test]  public void When\_User\_ProposeTo\_User()  {  IUser Peter = ObjectFactory.GetInstance<IUser>();  IUser Patty = ObjectFactory.GetInstance<IUser>();  Peter.ProposeTo(Patty);  Peter.Marry(Patty);  Peter.Name = "Peter";  Patty.Name = "Patty";  Assert.IsTrue(Peter.Married, "Failed: Peter.Married");  Assert.IsTrue(Peter.Age < 21, "Failed: Peter.Age < 21");  Assert.IsTrue(Patty.Married, "Failed: Patty.Married");  Assert.IsTrue(Patty.Age < 21, "Failed: Patty.Age < 21");  Assert.IsTrue(Peter.Spouse == Patty, "Failed: Peter.Spouse == Patty");  Assert.IsTrue(Patty.Spouse == Peter, "Failed: Patty.Spouse == Peter");  } |

**Test with If/Else Statements**

|  |
| --- |
| DSL |
| When a @**User**(user) ~**Name** is ("Logan"),  if @**User**(user) ~**IsDead** is (false) and @**User**(user) ~**CreatedDate** is later than ("22/04/2000") then @**User**(user) should have ~**IsActive** as (true)and @**User**(user) ~**Name** should contain ("Log")  else if @**User**(user) ~**CreatedDate** is earlier than ("22/04/2010") then @**User**(user) should have less than 3 ~**Children**  else @**User**(user) should have more than 3 ~**Children** |
| C# |
| [Test]  public void When\_User\_Name\_Is\_Set()  {  IUser user = ObjectFactory.GetInstance<IUser>();  user.Name = "Logan";  var dateTime20 = DateTime.Parse("22/04/2010");  var dateTime21 = DateTime.Parse("22/04/2000");  if (!user.IsDead && user.CreatedDate > dateTime21)  {  Assert.IsTrue(user.IsActive, "Failed: user.IsActive");  Assert.IsTrue(user.Name.Contains("Log"),"Failed:  user.Name.Contains(\"Log\")");  }  else if (user.CreatedDate < dateTime20)  {  Assert.IsTrue(user.Children.Count < 3, "Failed: user.Children.Count < 3");  }  else  {  Assert.IsTrue(user.Children.Count > 3, "Failed: user.Children.Count > 3");  }  } |