Fluent NHibernate wiki

Persistence specification testing

From Fluent NHibernate

You can quickly test your mappings using the `PersistenceSpecification<T>` class. Let's start with a simple Employee mapping:

```
public class Employee
{
    public virtual int Id { get; private set; }
    public virtual string FirstName { get; set; }
    public virtual string LastName { get; set; }
}

public class EmployeeMap : ClassMap<Employee>
{
    public EmployeeMap()
    {
        Id(x => x.Id);
        Map(x => x.FirstName);
        Map(x => x.LastName);
    }
}
```

To test this mapping:

```
[Test]
public void CanCorrectlyMapEmployee()
{
    new PersistenceSpecification<Employee>(session)
        .CheckProperty(c => c.Id, 1)
        .CheckProperty(c => c.FirstName, "John")
        .CheckProperty(c => c.LastName, "Doe")
        .VerifyTheMappings();
}
```

This test will :

- create an Employee instance
- insert the Employee into the database
- retrieve the record into a new Employee instance
- verify the retrieved Employee matches the original

Testing references

Let's consider an Employee mapping that includes a reference to a Store:

```
public class EmployeeMap : ClassMap<Employee>
{
```

```
    public EmployeeMap()
    {
        Id(x => x.Id);
        Map(x => x.FirstName);
        Map(x => x.LastName);
        References(x => x.Store);
    }
}
```

When verifying mappings, the Object.Equals() method is used to compare the retrieved values to the inserted values. This works as expected for primitive types and types that override Object.Equals(). For all other types, such as our Store class, the default implementation of Object.Equals() checks that both values point to the same instance of the object. We will always have different instances because the test flushes and clears the session cache after inserting the record. Instead, we want to verify the correct Store was loaded by comparing the Store's Id (and possibly other properties).

The PersistenceSpecification constructor allows us to pass in an IEqualityComparer to customize how each value is compared.

```
[Test]
public void CanCorrectlyMapEmployee()
{
    new PersistenceSpecification<Employee>(session, new CustomEqualityComparer())
        .CheckProperty(c => c.Id, 1)
        .CheckProperty(c => c.FirstName, "John")
        .CheckProperty(c => c.LastName, "Doe")
        .CheckReference(c => c.Store, new Store() {Name = "MyStore"})
        .VerifyTheMappings();
}

public class CustomEqualityComparer: IEqualityComparer
{
    public bool Equals(object x, object y)
    {
        if (x == null || y == null)
        {
            return false;
        }
        if (x is Store && y is Store)
        {
            return ((Store) x).Id == ((Store) y).Id;
        }
        return x.Equals(y);
    }

    public int GetHashCode(object obj)
    {
        throw new NotImplementedException();
    }
}
```

Note that, in order for CheckReferences to work, you either need to pass an already persistence entity, or have cascading on the entities you're testing. Otherwise, the test will fail complaining "object references an unsaved transient instance".

Obviously, the sample provided here won't work because the Store object is transient so here you'll need Cascading to make the test pass (which is omitted for brevity).

Retrieved from "http://wiki.fluentnhibernate.org/Persistence_specification_testing"

Page Discussion View source History