Unit Testing

Unit Testing A Business Layer Method

Arthur Clarkson

4th September 2024

A unit test is designed to evaluate specific software components or methods, often referred to as a "unit of work." These tests focus solely on code that is under the developer's direct control and avoid testing external infrastructure. Infrastructure concerns, such as interactions with databases, file systems, or network resources, are not part of unit testing.

After reading the specification I came up with this method to create holiday calendar events:

```
/// <summary>
/// Creates a CalendarEvent object based off a user's holiday details.
/// If the holiday is longer than one day, it will add a range to the description. E.g: "15th Jul - 16th
Jul"
/// </summary>
/// <returns>
/// Calendar event spanning over holidayStartDate-holidayEndDate with a description of "{username} HOL
{durationType.DisplayEnum()} {holidayDurationDescriptionSection}" (e.g. "aclarkson HOL All Day 15th Jul -
16th Jul")
/// </returns>
public CalendarEvent CreateHolidayCalendarEvent
(
          DateTime holidayStartDate,
          DateTime holidayEndDate,
          HolidayEntryItemDurationEnum durationType,
          int userId,
          string username,
          int calendarId
)
{
          // Setting startDate to smallest date and endDate to largest.
          DateTime tempDate = holidayStartDate;
          holidayStartDate = holidayStartDate.Min(holidayEndDate);
          holidayEndDate = tempDate.Max(holidayEndDate);

          double durationDifferenceInDays = (holidayEndDate - holidayStartDate).Days;

          bool holidayDatesAreInSameWeek = holidayStartDate.IsInSameWeekAs(holidayEndDate);
          bool durationIsForMoreThanOneDay = durationDifferenceInDays >= 1;

          string holidayDurationSection = "";

          if (durationIsForMoreThanOneDay)
          {
                        holidayDurationSection = holidayDatesAreInSameWeek ?
                                  $"{holidayStartDate.DisplayDay()} - {holidayEndDate.DisplayDay()}" :
// e.g. Mon - Fri
                                  $"{holidayStartDate.DisplayDayOfMonth()} -
{holidayEndDate.DisplayDayOfMonth()}"; // e.g. 17 Jul - 23 Jul
          }

          string holidayDurationTypeSection = "";
          bool descriptionRequiresDurationType = durationType is not
HolidayEntryItemDurationEnum.AllDay;
          if(descriptionRequiresDurationType)
          {
                        holidayDurationTypeSection = durationType.DisplayEnum();
          }

          string description = "";
          string[] descriptionParameters = new string[] { username,
TimesheetController.KnownJobcodes.HOLIDAY, holidayDurationTypeSection, holidayDurationSection };

          foreach(string parameter in descriptionParameters)
          {
                  description += $"{parameter} ";
          }

          description = description.Trim();

          CalendarEvent calendarEvent = new()
          {
                        CalendarId = calendarId,
                        Description = description,
                        Type = CalendarEventType.Holiday,
                        StartDate = holidayStartDate,
                        EndDate = holidayEndDate,
                        Status = 1,
                        DateEntered = DateTime.UtcNow,
                        DateModified = DateTime.UtcNow,
                        PrimaryContactUserId = userId
          };

          InsertCalendarEvent(calendarEvent);

          return calendarEvent;
}
```

With there being many permutations with this method, I decided to unit test it.

Unit testing is fully automated, and are matched against the specification, they make sure new and amended code gives desired outputs, they check if there are any inputs which would cause a run-time error, making the test fail.

Using C#'s NUnit testing framework I came up with this unit test and it's test data:

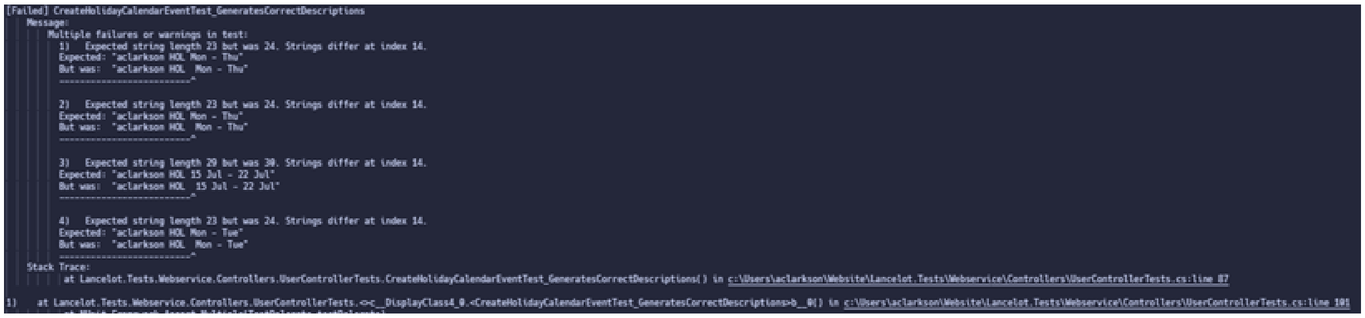
```
[Test]
public void CreateHolidayCalendarEventTest_GeneratesCorrectDescriptions()
{
        int userId = 2120;
        string username = "aclarkson";
        int calendarId = 1;

        var testCases = new[]
        {
                new
                {
                        StartDate = new DateTime(2024, 07, 15),
                        EndDate = new DateTime(2024, 07, 18),
                        Duration = HolidayEntryItemDurationEnum.AllDay,
                        ExpectedDescription = "aclarkson HOL Mon - Thu"
                },
                new
                {
                        StartDate = new DateTime(2024, 07, 15),
                        EndDate = new DateTime(2024, 07, 15),
                        Duration = HolidayEntryItemDurationEnum.Morning,
                        ExpectedDescription = "aclarkson HOL AM"
                },
                new
                {
                        StartDate = new DateTime(2024, 07, 15),
                        EndDate = new DateTime(2024, 07, 18),
                        Duration = HolidayEntryItemDurationEnum.AllDay,
                        ExpectedDescription = "aclarkson HOL Mon - Thu"
                },
                new
                {
                        StartDate = new DateTime(2024, 07, 15),
                        EndDate = new DateTime(2024, 07, 18),
                        Duration = HolidayEntryItemDurationEnum.Afternoon,
                        ExpectedDescription = "aclarkson HOL PM Mon - Thu"
                },
                new
                {
                        StartDate = new DateTime(2024, 07, 15),
                        EndDate = new DateTime(2024, 07, 22),
                        Duration = HolidayEntryItemDurationEnum.AllDay,
                        ExpectedDescription = "aclarkson HOL 15 Jul - 22 Jul"
                },
                new
                {
                        StartDate = new DateTime(2024, 07, 15),
                        EndDate = new DateTime(2024, 07, 16),
                        Duration = HolidayEntryItemDurationEnum.AllDay,
                        ExpectedDescription = "aclarkson HOL Mon - Tue"
                }
        };

        Assert.Multiple(() =>
        {
                foreach (var testCase in testCases)
                {
                        var calendarEvent = _calendarController.CreateHolidayCalendarEvent(
                                testCase.StartDate,
                                testCase.EndDate,
                                testCase.Duration,
                                userId,
                                username,
                                calendarId
                        );

                        string actualDescription = calendarEvent.Description;
                        Assert.That(actualDescription, Is.EqualTo(testCase.ExpectedDescription));
                }
        });
}
```

After running the test, I get failure on the following cases:



```
[Failed] CreateHolidayCalendarEventTest_GeneratesCorrectDescriptions
    Message:
        Multiple failures or warnings in test:
          1)   Expected string length 23 but was 24. Strings differ at index 14.
          Expected: "aclarkson HOL Mon - Thu"
          But was:  "aclarkson HOL  Mon - Thu"
          -------------------------^

          2)   Expected string length 23 but was 24. Strings differ at index 14.
          Expected: "aclarkson HOL Mon - Thu"
          But was:  "aclarkson HOL  Mon - Thu"
          -------------------------^

          3)   Expected string length 29 but was 30. Strings differ at index 14.
          Expected: "aclarkson HOL 15 Jul - 22 Jul"
          But was:  "aclarkson HOL  15 Jul - 22 Jul"
          -------------------------^

          4)   Expected string length 23 but was 24. Strings differ at index 14.
          Expected: "aclarkson HOL Mon - Tue"
          But was:  "aclarkson HOL  Mon - Tue"
          -------------------------^
    Stack Trace:
           at Lancelot.Tests.Webservice.Controllers.UserControllerTests.CreateHolidayCalendarEventTest_GeneratesCorrectDescriptions() in c:\Users\aclarkson\Website\Lancelot.Tests\Webservice\Controllers\UserControllerTests.cs:line 87

1)    at Lancelot.Tests.Webservice.Controllers.UserControllerTests.<>c__DisplayClass4_0.<CreateHolidayCalendarEventTest_GeneratesCorrectDescriptions>b__0() in c:\Users\aclarkson\Website\Lancelot.Tests\Webservice\Controllers\UserControllerTests.cs:line 101
```

It looked like it was trying to add null/empty parameters to the description text, adding a random space to the string, due to this code:

```
string description = "";
string[] descriptionParameters = new string[] { username, TimesheetController.KnownJobcodes.HOLIDAY,
holidayDurationTypeSection, holidayDurationSection };

foreach(string parameter in descriptionParameters)
{
  description += $"{parameter} ";
}
```

I decided to add the following null check:

```
string description = "";
string[] descriptionParameters = new string[] { username, TimesheetController.KnownJobcodes.HOLIDAY,
holidayDurationTypeSection, holidayDurationSection };

foreach(string parameter in descriptionParameters)
{
        bool parameterIsNotEmpty = false == string.IsNullOrWhiteSpace(parameter);
        if(parameterIsNotEmpty)
        {
                  description += $"{parameter} ";
        }
}
```

After running the test again, it managed to pass all cases.