



Освен прихващането на изключения какви най-често проверки се правят за методите, когато създаваме unit tests?



Когато създаваме **Unit тестове** за методи, освен прихващането на изключения, има различни видове проверки, които се използват, за да се гарантира, че методите работят правилно в различни сценарии. Ето някои от най-често срещаните проверки, които се правят при създаването на Unit тестове:

1. Проверки за очакван изход (Assert.AreEqual, Assert.AreNotEqual)

Това е най-често срещаната проверка – проверяваш дали резултатът от метода е очакваният.

csharp

Копиране на код

```
[Test]
public void Add_ShouldReturnSum_WhenGivenValidNumbers()
{
    int result = calculator.Add(2, 3);
    Assert.AreEqual(5, result);
}
```

Тук проверяваме дали методът `Add` връща правилната сума.

2. Проверки на булеви условия (Assert.IsTrue, Assert.IsFalse)

Тези проверки се използват, за да се провери дали дадено условие е вярно или невярно.

csharp

Копиране на код

```
[Test]
public void IsAdult_ShouldReturnTrue_WhenAgeIs18OrMore()
{
    bool result = person.IsAdult(18);
    Assert.IsTrue(result);
}

[Test]
public void IsAdult_ShouldReturnFalse_WhenAgeIsLessThan18()
{
    bool result = person.IsAdult(17);
    Assert.IsFalse(result);
}
```

Тук се проверява дали методът за проверка на възрастта връща вярно за 18+ години и невярно за по-малка възраст.

3. Проверки за референтни типове (Assert.IsNull, Assert.IsNotNull)

Тези проверки се използват, за да се гарантира, че даден обект е или `null`, или не е `null`.

csharp

Копиране на код

```
[Test]
public void GetUser_ShouldReturnNull_WhenUserDoesNotExist()
{
    var user = userService.GetUser("nonexistent");
    Assert.IsNull(user);
}

[Test]
public void GetUser_ShouldReturnUser_WhenUserExists()
{
    var user = userService.GetUser("existingUser");
    Assert.IsNotNull(user);
}
```



Тези проверки са важни при работа с методи, които могат да върнат `null`, когато обект не съществува, или да върнат валиден обект при успешни заявки.

4. Проверки за съдържание на колекции (Assert.Contains, Assert.DoesNotContain, CollectionAssert)

Когато методът връща колекция (например масив или списък), проверките могат да гарантират, че определени елементи са или не са част от резултата.

csharp

Копиране на код


```
[Test]
public void GetActiveUsers_ShouldContainActiveUser()
{
    var users = userService.GetActiveUsers();
    Assert.Contains("activeUser", users);
}

[Test]
public void GetActiveUsers_ShouldNotContainInactiveUser()
{
    var users = userService.GetActiveUsers();
    Assert.DoesNotContain("inactiveUser", users);
}
```

5. Проверки на дължина или брой на елементи (Assert.AreEqual, CollectionAssert.Count)

Може да провериш дали броят на елементите в резултатната колекция е правилен.

csharp


 Копиране на код

```
[Test]
public void GetAllUsers_ShouldReturnCorrectNumberOfUsers()
{
    var users = userService.GetAllUsers();
    Assert.AreEqual(5, users.Count);
}
```

6. Мокване на зависимости и проверка на повиквания (Moq или други mocking библиотеки)

Когато тестваш методи, които имат зависимости от други класове или услуги, често се използват mock обекти, за да се проверят взаимодействията между тях.

csharp

 Копиране на код

```
[Test]
public void SaveUser_ShouldCallRepositorySaveOnce()
{
    var mockRepo = new Mock<IUserRepository>();
    var userService = new UserService(mockRepo.Object);

    userService.SaveUser(new User());

    mockRepo.Verify(repo => repo.Save(It.IsAny<User>()), Times.Once);
}
```




- Пропускаме на този етап

7. Гранични условия (Boundary Testing)

Тестването на граничните стойности е критично за гарантиране, че методите работят правилно при минимални, максимални или извън нормалните стойности.

csharp


 Копиране на код

```
[Test]
public void CalculateDiscount_ShouldReturnZero_WhenAmountIsLessThanMinimum()
{
    decimal result = calculator.CalculateDiscount(0);
    Assert.AreEqual(0, result);
}
```

8. Тестване на резултати от асинхронни операции

Когато се работи с асинхронни методи, трябва да се проверява правилното връщане на `Task` и успешното завършване на асинхронните операции.

csharp

 Копиране на код

```
[Test]
public async Task GetDataAsync_ShouldReturnData_WhenCalled()
{
    var result = await dataService.GetDataAsync();
    Assert.IsNotNull(result);
}
```

Обобщение:

Тези проверки гарантират, че методите работят коректно в широк спектър от ситуации. Те помагат за валидирането както на очаквания резултат, така и на взаимодействията и поведението на методите, което е от съществено значение за надеждното поведение на приложенията.