**ЛАБОРАТОРНАЯ РАБОТА №3**

по дисциплине «Тестирование программного обеспечения»

**«**Автоматизация модульного тестирования**»**

**Цель работы**: закрепление практических навыков по проведению автоматизации модульного тестирования.

**Задание:** провести автоматизацию разработанного ранее программного кода согласно описанной последовательности.

Автор: студент гр. 10702318 Чваньков А.

Проверил: Белова С.В.

Оглавление

[1. Класс Controller. 3](#_Toc67304670)

[1.1 Метод Search. 3](#_Toc67304671)

[2. Класс Controller. 3](#_Toc67304672)

[2.1 Метод CheekTheSamePerson. 3](#_Toc67304673)

[3. Класс Controller. 6](#_Toc67304674)

[3.1 Метод indexSearch 6](#_Toc67304675)

[4. Результаты тестирования. 9](#_Toc67304678)

1. Класс Controller.  
   Для тестирования данного модуля используется NUnit.

1.1 Метод Search.

Тело метода:

public IEnumerable<Magician> Search(string character)

{

List<Magician> result = new List<Magician>();

for (var i = 0; i < data.Count; i++)

{

if (data[i].ToString().ToLower().Contains(character.ToLower()) && character.Contains(",") == false)

{

result.Add(data[i]);

}

}

return result;

}

Тесты:

[TestMethod]

public void SearchTest\_ReturnValidData()

{

var controller = Controller.GetInstance();

controller.Create(new Magician

{

Id = 1,

KindOfMagic = KindOfMagic.Air,

HitPoints = 200,

Name = "Black Overlord",

Power = 1000,

Speed = 50

});

controller.Create(new Magician

{

Id = 2,

KindOfMagic = KindOfMagic.Fire,

HitPoints = 400,

Name = "Negativ",

Power = 2000,

Speed = 100

});

var expectedResult = new List<Magician>

{

new Magician

{

Id = 1,

KindOfMagic = KindOfMagic.Air,

HitPoints = 200,

Name = "Black Overlord",

Power = 1000,

Speed = 50

}

};

var actualResult = controller.Search("Air").ToList();

Assert.AreEqual(expectedResult[0], actualResult[0]);

}

2. Класс Controller.  
Для тестирования этого модуля (и всех последующих) используется MSTest.

2.1 Метод CheekTheSamePerson.

Тело метода:

public void CheekTheSamePerson()

{

for (var i = 0; i < data.Count; i++)

{

if (i + 1 >= data.Count) continue;

for (var j = i + 1; j < data.Count; j++)

{

if (data[i].Equals(data[j]))

{

data.RemoveAt(i);

}

}

}

}

Тесты:

[TestMethod]

public void CheekTheSamePersonTest\_NoSamePersons()

{

var controller = Controller.GetInstance();

controller.Create(new Magician

{

Id = 1,

KindOfMagic = KindOfMagic.Air,

HitPoints = 200,

Name = "Black Overlord",

Power = 1000,

Speed = 50

});

controller.Create(new Magician

{

Id = 2,

KindOfMagic = KindOfMagic.Fire,

HitPoints = 400,

Name = "Negativ",

Power = 2000,

Speed = 100

});

controller.Create(new Magician

{

Id = 2,

KindOfMagic = KindOfMagic.Fire,

HitPoints = 400,

Name = "Negativ",

Power = 2000,

Speed = 100

});

var expectedResult = new List<Magician>

{

new Magician

{

Id = 1,

KindOfMagic = KindOfMagic.Air,

HitPoints = 200,

Name = "Black Overlord",

Power = 1000,

Speed = 50

},

new Magician

{

Id = 2,

KindOfMagic = KindOfMagic.Fire,

HitPoints = 400,

Name = "Negativ",

Power = 2000,

Speed = 100

}

};

controller.CheekTheSamePerson();

var actualResult = controller.GetAll().ToList();

for (var i = 0; i < actualResult.Count; i++)

{

Assert.AreEqual(expectedResult[i], actualResult[i]);

}

}

3. Класс Controller.

3.1 Метод indexSearch  
Метод:

public string indexSearch(int id)

{

string result = " ";

for (var i = 0; i < data.Count; i++)

{

if (data[i].Id == id)

{

return data[i].ToString();

}

else

{

return "Magician not found" ;

}

}

return result;

}

Тесты:

[TestMethod]

public void IndexSearchTest\_FindMagician()

{

var controller = Controller.GetInstance();

controller.Create(new Magician

{

Id = 1,

KindOfMagic = KindOfMagic.Air,

HitPoints = 200,

Name = "Black Overlord",

Power = 1000,

Speed = 50

});

controller.Create(new Magician

{

Id = 2,

KindOfMagic = KindOfMagic.Fire,

HitPoints = 400,

Name = "Negativ",

Power = 2000,

Speed = 100

});

var expectedResult = new List<Magician>

{

new Magician

{

Id = 1,

KindOfMagic = KindOfMagic.Air,

HitPoints = 200,

Name = "Black Overlord",

Power = 1000,

Speed = 50

},

};

var actualResult = controller.indexSearch(1);

for (var i = 0; i < expectedResult.Count; i++)

{

Assert.AreEqual(expectedResult[i].ToString(), actualResult);

}

}

[TestMethod]

public void IndexSearchTest\_MagicianNotFound()

{

var controller = Controller.GetInstance();

controller.Create(new Magician

{

Id = 1,

KindOfMagic = KindOfMagic.Air,

HitPoints = 200,

Name = "Black Overlord",

Power = 1000,

Speed = 50

});

controller.Create(new Magician

{

Id = 2,

KindOfMagic = KindOfMagic.Fire,

HitPoints = 400,

Name = "Negativ",

Power = 2000,

Speed = 100

});

var expectedResult = "Magician not found";

var actualResult = controller.indexSearch(3);

Assert.AreEqual(expectedResult, actualResult);

}

[TestMethod]

public void IndexSearchTest\_NoMagicianInDatabase()

{

var controller = Controller.GetInstance();

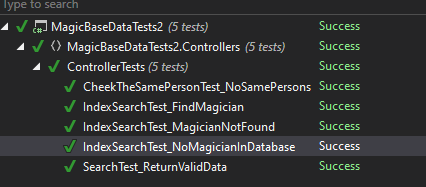
var expectedResult = " ";

var actualResult = controller.indexSearch(3);

Assert.AreEqual(expectedResult, actualResult);

}

1. Результаты тестирования.



Вывод.

Платформа тестирования Microsoft платная, но как только вы заплатили за весь пакет, вы получаете модульный тест, тестовый раннер и покрытие кода в одном пакете, что очень удобно.

Если вы переключитесь на NUnit, вам придется либо отказаться от интеграции VS, либо получить тестовый раннер, такой как TestDriven.NET или Resharper. Но они оба требуют лицензий для коммерческого использования. Кроме того, эти инструменты не имеют покрытия кода, поэтому вы должны получить что-то вроде NCover. А это, опять же, лицензия на коммерческое использование.