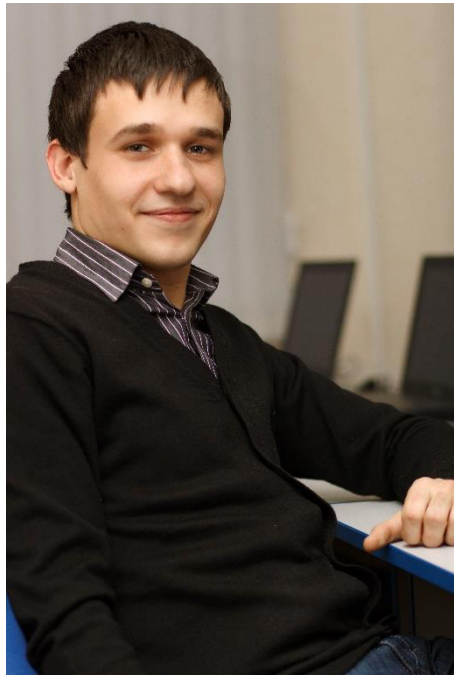
Разработка через тестирование

Использование Stub объектов для Unit тестов

Разработка через тестирование

Автор курса



Давид Бояров

MCID: 9778145

Разработка через тестирование

После урока обязательно

Повторите этот урок в видео формате на ITVDN.com

Доступ можно получить через руководство вашего учебного центра

Проверьте как Вы усвоили данный материал на TestProvider.com

Тема

Использование Stub объектов для Unit тестов

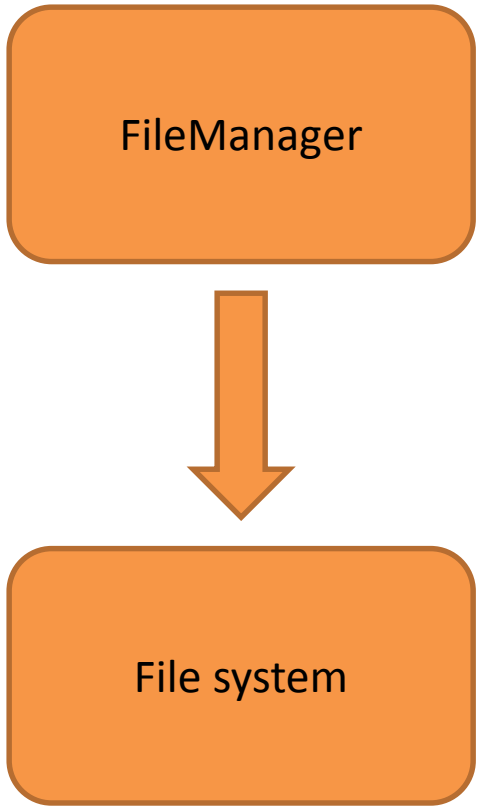
Внешняя зависимость

Определение

Внешняя зависимость – объект в системе, с которым взаимодействует тестируемый код, и который невозможно контролировать (например, файловая система, потоки, память, службы и т. д.).

Внешняя зависимость

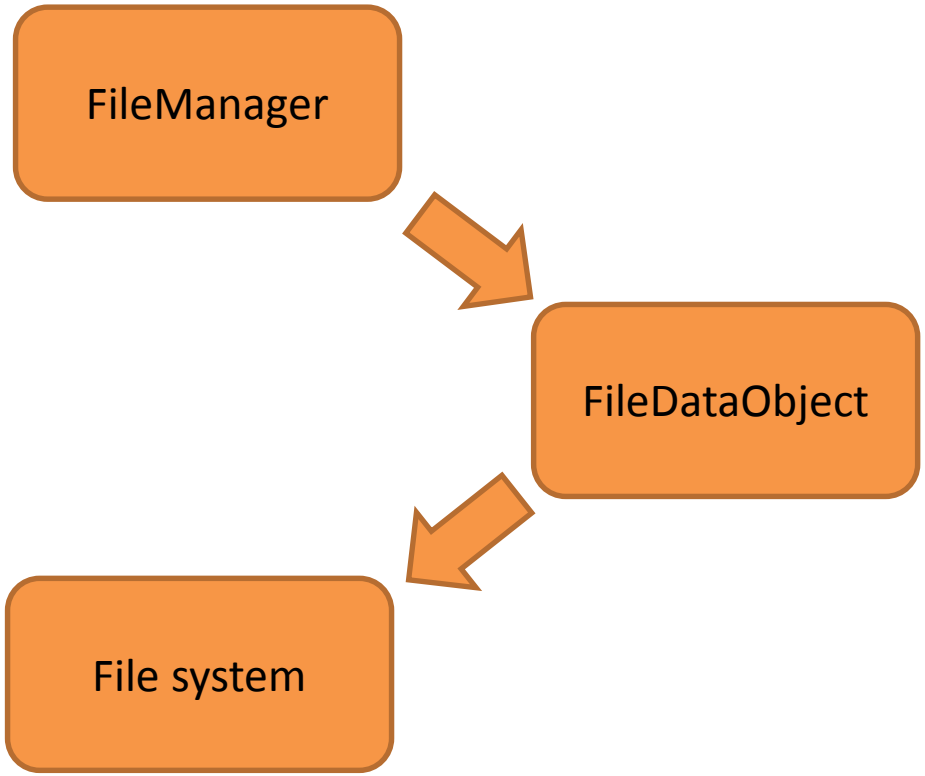
Проблема



Тест для FileManager’а будет зависить от файловой системы, сответственно будет является интеграционным, так как мы будет тестировать не только FileManager, но и файловую систему (наличие файла, правильность формата файла и т.д.).

Внешняя зависимость

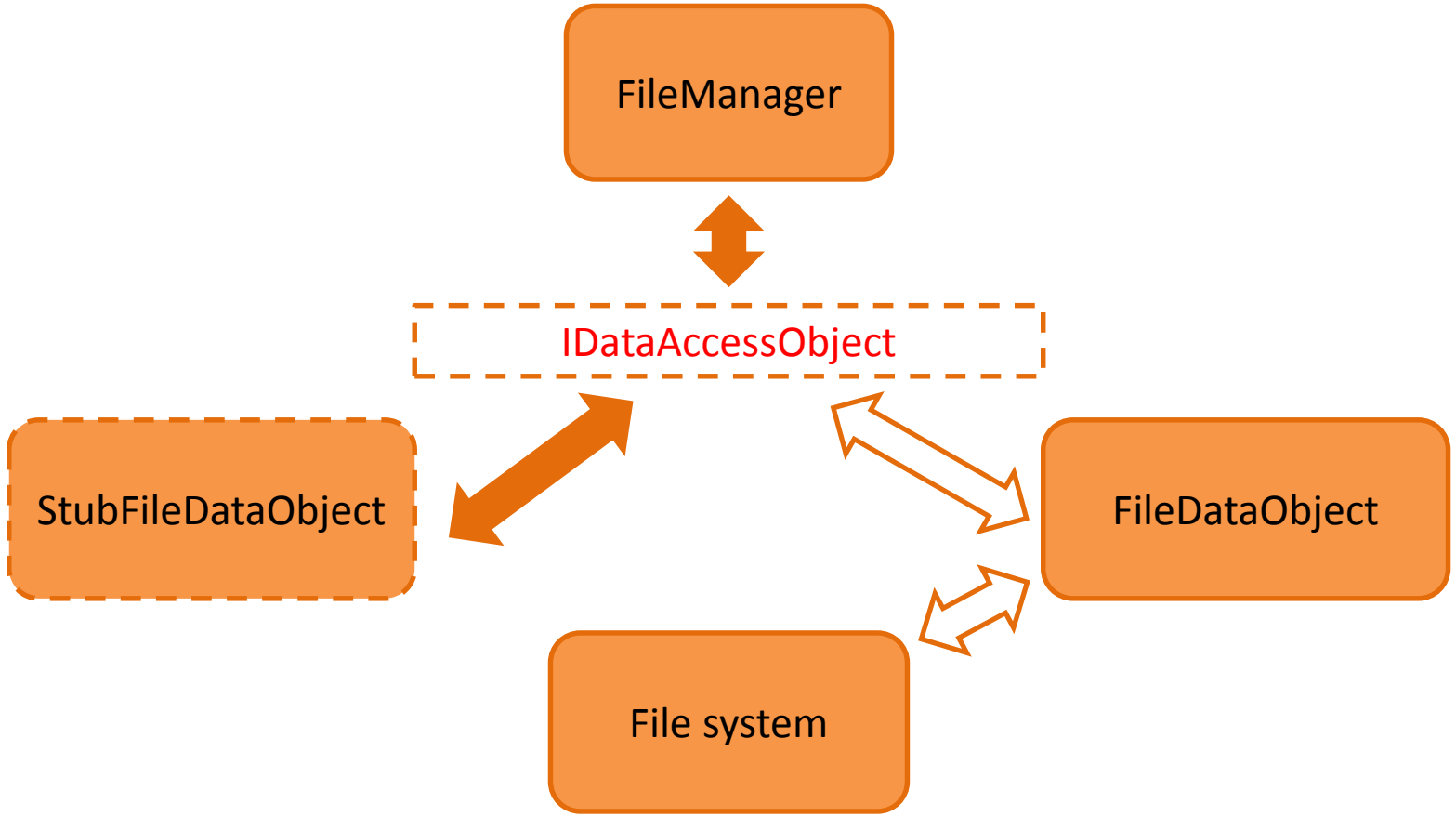
Проблема



Введение косвенного слоя для избежания прямой зависимости от файловой системы. Код, который работает с файловой системой разделяется на FileDataObject, который в будущем будет заменен с stub объектом в тесте.

Создание stub объекта

Новый интерфейс позволит сделать объектную модель, чтобы абстрагироваться от операций класса FileDataObject, и создать тест с заглушкой.



Что такое Stub объект?

Определение

Stub-объект (заглушка) – это управляемая замена существующих зависимостей в системе. Stub-объект позволяет тестировать код без использования внешних зависимостей.

Проблемы при проектировании приложений

Зависимости

Dependency Injection (DI)

Внедрение зависимости

Dependency Injection – паттерн, описывающий технику внедрения внешней зависимости программному компоненту.

Внедрение зависимости

Способы внедрения зависимостей:

1. Внедрение через конструктор.
2. Внедрение через свойство.
3. Внедрение через интерфейсы.

Способы создания экземпляров зависимостей:

1. Через контейнеры.
2. С помощью Ninject.

Внедрение через конструктор

Конструктор, через который будет внедрена зависимость:

```
public FileManager(IDataAccessObject dataAccessObject)
{
    this.dataAccessObject = dataAccessObject;
}

...
```

Вызов метода Stub объекта:

```
dataAccessObject.GetFiles();
```

Внедрение через свойство

Свойство, через которое будет внедрена зависимость:

```
public IDataAccessObject DataAccessObject
{
    set { dataAccessObject = value; }
    get {
         if (dataAccessObject == null)
         {
             throw new MemberAccessException("DataAccessObject has
                                              not been initialized.");
         }
         return dataAccessObject;
        }
}

...
```

Вызов метода Stub объекта:

```
DataAccessObject.GetFiles();
```

Внедрение через интерфейс

Интерфейс, через который будет внедрена зависимость:

```
public bool FindLogFile(string fileName, IDataAccessObject dataAccessObject)
{
    Вызов метода Stub объекта:
    dataAccessObject.GetFiles();
}
```

Dependency Injection контейнеры

DI Containers

- Castle Windsor (AltDotNet)

http://www.castleproject.org/

- Unity (Microsoft P&P)

http://www.codeplex.com/unity

- Ninject (open source)

http://ninject.org

- StructureMap (AltDotNet)
- Много других (LinFu, например)

Использование DI контейнеров

Unity

Создание DI контейнера:

```
IUnityContainer container = new UnityContainer();
```

Регистрирование типов интерфейсов и классов, которые их реализуют для инстансирования зависимостей:

```
container.RegisterType<IDataAccessObject,
StubFileDataObject>();
```

Создание объекта типа FileManager и внедрение всех зависимостей в соответствии с зарегистрированными в контейнере типами:

```
FileManager manager = container.Resolve<FileManager>();
```

Использование DI контейнеров

Ninject

Определение конфигурации:

```
class ConfigFileObjectData : NinjectModule
{
    public override void Load()
    {

this.Bind<IDataAccessObject>().To<StubFileDataObject>();
    }
}
```

Использование, например:

```
IKernel ninjectKernel = new StandardKernel(new
ConfigFileObjectData());

FileManager manager = ninjectKernel.Get<FileManager>();
```

Атрибут InternalsVisibleTo

InternalsVisibleToAttribute

Делает internal типы видимыми для заданной сборки:

```
[assembly: InternalsVisibleTo("001_TestOperations")]
```

Как правило типы и элементы с модификатором доступа internal доступны только в сборке, в которой они определены.
Атрибут InternalsVisibleToAttribute делает их также видимыми для типов в указанной сборке, которая называется "дружественная сборка".

Смотрите наши уроки в видео формате

ITVDN.com

Посмотрите этот урок в видео формате на образовательном портале ITVDN.com для закрепления пройденного материала.

Все курсы записаны сертифицированными тренерами, которые работают в учебном центре CyberBionic Systematics

ITVDN
IT VIDEO DEVELOPERS NETWORK

Проверка знаний

TestProvider.com

TestProvider – это online сервис проверки знаний по информационным технологиям. С его помощью Вы можете оценить Ваш уровень и выявить слабые места. Он будет полезен как в процессе изучения технологии, так и общей оценки знаний IT специалиста.

После каждого урока проходите тестирование для проверки знаний на TestProvider.com

Успешное прохождение финального тестирования позволит Вам получить соответствующий Сертификат.

TestProvider

Q&A

Информационный видеосервис для разработчиков программного обеспечения

ITVDN

IT VIDEO DEVELOPERS NETWORK