Использование Stub объектов для Unit тестов

№ урока:	2	**Курс:**	Test Driven Development

Средства обучения:	Компьютер с установленной Visual Studio NUnit Framework NUnit Adapter (optional)

Обзор, цель и назначение урока

- Понимать и определять зависимости.
- Научиться использовать Stub объекты.
- Научиться использовать DI контейнеры.

Изучив материал данного занятия, учащийся сможет:

- Создавать Unit тесты, используя Stub объекты.
- Использовать различные техники внедрения зависимости.
- Решать проблемы инкапсуляции при внедрении зависимости.

Содержание урока

1. Понятие внешней зависимости.
2. Проблемы тестирования при появлении внешней зависимости.
3. Понятие Stub объекта
4. Создание и использование Stub-объектов.
5. Паттерн внедрения зависимости.
6. Различные техники внедрения зависимости
7. Понятие DI контейнера.
8. Использование DI контейнеров на примере Unity и Ninject.
9. Решение проблемы инкапсуляции при внедрении зависимости.

Резюме

- **Внешняя зависимость** – объект в системе, с которым взаимодействует тестируемый код, и который невозможно контролировать (например, файловая система, потоки, память, службы и т. д.).
- Как правило, внешняя зависимость появляется в следующих случаях:
    1. При создании объекта явно указывается класс.
    2. Зависимость от аппаратных/программных платформ.
    3. Зависимость от представления или реализации объекта.
    4. Зависимость от алгоритмов.
- Сильная зависимость порождает следующие проблемы. Систему сложно:
    1. поддерживать
    2. Расширять
    3. Понимать
    4. Тестировать
- **Dependency Injection** – паттерн, описывающий технику внедрения внешней зависимости программному компоненту.
- Преимущества использования **Dependency Injection:**

CyberBionic systematics

CyberBionic Systematics ® 2015
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine

t. +380 (44) 361-8473
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com

1. Разделение конфигурирования и использования объектов
2. Уменьшается связь между объектами. Конкретные объекты проще заменить.
3. Увеличение мобильности модулей
4. Систему проще сопровождать и тестировать.

- **Способы внедрения зависимости:**
  1. Внедрение через интерфейс.
  2. Внедрение через свойство. Используется, если зависимость имеет опциональный характер.
  3. Внедрение через конструктор. Проблемно использовать, если для правильной работы тестируемого класса требуется несколько Sub-объектов. В этом случае приходится создавать или множество конструкторов, или же один конструктор с множеством параметров.
- **Inversion of Control (IoC)** – абстрактный принцип, описывающий способы написания слабосвязанного кода. Dependency Injection – один из способов реализации данного принципа.
- **DI container** – набор объектов, позволяющая упростить и автоматизировать процесс написания кода с использованием принципа Inversion of Control.
- **Stub-объект (заглушка)** – это управляемая замена существующих зависимостей в системе. Stub-объект позволяет тестировать код без использования внешних зависимостей.
- Для решения проблем инкапсуляции членов, созданных исключительно в целях тестирования можно использовать директивы препроцессора **#if, #elif, #endif** а так же атрибут **InternalsVisibleToAttribute**.
- 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".

Закрепление материала

1. Что такое внешняя зависимость?
2. Какие недостатки внешней зависимости вы знаете?
3. Что собой представляет паттерн Dependency Injection?
4. Какие преимущества использования паттерна Dependency Injection?
5. Какие способы внедрения зависимости вы знаете?
6. Что такое Stub объекты? Для чего они предназначены?

CyberBionic Systematics ® 2015
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine
t. +380 (44) 361-8473
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com