Внедрение зависимостей (Dependency Injection)

Внедрение зависимостей (Dependency Injection)

Тема лекции

Внедрение зависимостей (Dependency Injection)

Цели и задачи

Знакомство с шаблоном проектирования Dependency Injection.

Изучение способов реализации Dependency Injection в ASP.NET MVC

Mark Seemann - Dependency Injection in .NET - Manning Publications Co., 2012. 499p.

(www.manning.com)

Класс Book

```
public class Book
    {
        public string author;   // автор
        public string title;    // название
    }
```

Класс управления хранилищем книг

```
public class BookStorage
    {
        private DataBase _db;
        public BookStorage()
        {
            _db = new DataBase();  // Создается объект DataBase
        }
        public void Store(string author, string title)
        {
            Book book = new Book(); // Создается объект Book
            book.author = author;    book.title = title;
            _db.Store(book);
        }   }
```

Добавить книгу в БД:

```
BookStorage _storage = new BookStorage();
_storage.Store(" Достоевский ", " Идиот" );
```

Внедрение зависимостей

В приведенном примере имеется **сильная связь** между классом BookStorage и классами Book и Database, т.к. внутри класса BookStorage создаются объекты других классов: Book и Database.

Внедрение зависимостей

Класс BookStorage невозможно протестировать независимо.

Т.к. объекты других классов: Book и Database создаются внутри класса BookStorage, невозможно подменить объекты этих классов макетами.

Модификация класса BookStorage

```
public class BookStorage
    {
        private DataBase db;
        public BookStorage(DataBase database)
        {
            db = database;
        }
    public void Store(Book book)
        {
            db.Store(book);
        }
    }
```

Добавить книгу в БД

```
var db = new DataBase();
var _storage = new BookStorage1(db);
var book = new Book {
        author = "Достоевский",
        title = "Идиот" };
_storage.Store(book);
```

Внедрение зависимостей

Все еще требуется для тестирования объект конкретного класса Database.
Добавим интерфейс:

```
public interface IDatabase
    {
        void Store (Book book);
    }
```

Внедрение зависимостей

Class Database: IDatabase
{ . . .}

```
public class BookStorage
{
        private IDatabase _db;
       public BookStorage(IDatabase db)
   {_db = db;}

    public void Store(Book book)
    {    _db.Store(book);  }
}
```

Внедрение зависимостей

Dependency Injection,
или "внедрение зависимости" – шаблон проектирования приложения, который предполагает, что каждый компонент ничего не знает о других компонентах, а взаимодействует с ними через интерфейсы.

Внедрение зависимостей

Это называется слабым связыванием (louse coupling) или **IoC** (**Inversion of Control** – инверсия управления) и упрощает процедуру тестирования и рефакторинга.
DI (Dependency Injection)

В литературе термины DI (dependency injection), IoC (inversion of control) означают один и тот же шаблон внедрения зависимостей

Внедрение зависимостей

IoC-контейнер — это библиотека, которая представляет собой фабрику. Как и любой другой класс фабрики, он отвечает за создание экземпляров объектов, но он также знает, как создавать экземпляры зависимостей объектов. Это означает, что мы можем попросить контейнер создать объект класса BookStorage, и он также создаст экземпляры всех зависимостей и передаст их в конструктор.

Dependency Injection

DI в .Net MAUI