Практическая работа № 3

Так, ладно. Легче не стало. Вторая работа сможет высосать душу из некоторых студентов. Но после неё точно ничего не страшно!

Dependency Injection (внедрение зависимостей)

Необходимо добиться того, чтобы объект, реализующий IRepository, внедрялся в объект класса BusinessLogic средствами IoC-контейнера (например Ninject) и представлял из себя *Singleton*.

Итак, нужно доработать лабораторную работу №2 следующим образом:

- В слое **View** создавать объект класса BusinessLogic необходимо через IoC-контейнер с целью автоматического внедрения зависимого объекта (репозитория) в объект BL.
- В слое BusinessLogic создаем сам контейнер.

Во второй лабораторной работе класс Бизнес-логики содержит ссылку на конкретную реализацию репозитория (EntityRepository или DapperRepository). Для того, чтобы убрать жесткую связь между двумя классами (классом бизнес-логики и репозитория) первым шагом определяем интерфейс IRepository (если ещё не сделали) и в классе бизнес-логики определяем конструктор с параметрами:

```
public class Logic
{
    public IRepository Repository { set; get; }

    public Logic(IRepository repository)
    {
        Repository = repository;
    }
}
```

Таким образом мы объявили зависимость бизнес-логики от объекта IRepository (а не от конкретного класса репозитория, как было раньше).

- Установим пакет Ninject.MVC5 через управление пакетами NuGet
- Зарегистрируем зависимости для всех репозиториев. Для этого создадим в библиотеке бизнес-логики новый класс SimpleConfigModule:

```
public class SimpleConfigModule : NinjectModule
{
    public override void Load()
    {
        Bind<IRepository<Employee>>().To<EntityRepository<Employee>>().InSingletonScope();
    }
}
```

Этим шагом мы устанавливаем сопоставление между интерфейсом-зависимостью и конкретным классом этого интерфейса.

На уровне представление (там, где мы ранее создавали экземпляр класса бизнес-логики), для управления зависимостями через Ninject создадим объект Ninject.IKernel с помощью встроенной реализации этого интерфейса - класса StandardKernel:

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

В этом случае создания экземпляра класса бизнес-логики необходимо писать так:

```
Logic bL = ninjectKernel.Get< Logic>();
```