**ОБЪЯСНЕНИЯ**

1) **DLL-библ**, реализ. модель данных  
- это реализация репозитория, кот. обесп. хранение д-х в json-файле (как №3)  
- это реализация репозитория, кот. обесп. хранени д-х в SQL-бд (как №5)  
репозиторий – реализация интерфейса IPhoneDictionary

**DDL/PhoneDictionarySql.cs:**\* есть некая модель Contact  
\* опред. контекст д-х  
\* опред. репоз. (PhoneDictionarySql) для управл. объектами этой модели

**Controllers:**

для подкл. репозитория к контроллеру исп. механизм Dependency Injection (позв. передать ссылку на объект репозитория как параметр контроллера

\*если бы сод. ссылка на PhoneDictionarySql , то контроллер **зависел бы** **от него**   
сам PhoneDictionarySql явл. бы зависимостью  
была бы *жесткая связь* – к-р ссылался бы на конкр. реализацию репозитория  
если м. захотеть динамически изменить один класс на другой

=> опред. ин-с IPhoneDictionaty в DDL  
\* параллельно м. опред другие классы репозитория (для json)  
теперь к-р ничего не знает и не зависит от конкр. реализации репоз-я  
(это наз. IoC – инверсия управления)

Вот мы объявили зависимость к-ра от IRepository  
теперь надо внедрить в к-р зависимость => исп. IoC-контейнеры (NinJect, AutoFac, Unity…)  
скач. Nuget

**AppStart/NinjectWebCommon.cs:  
CreateKernel() :** 1. создать объект Ninject.IKernel

**DI/NIConfig.cs:**

2. устан отношения между ин-сами и их реализациями (Bind)  
 это указ, что объекты IPhoneDictionary будут рассм. как PhoneDictionarySql  
 (?):repo = ninjectKernel.Get<IRepository>(); - созд. объект ин-са ч/Get (здесь будет созд. экземпляр PhoneDictSql)

1) тип внедрения: кот. создает свой объект репозитория для к. внедрения  
2) тип внедрения: объект реп. создается для к. потока выполнения  
3) тип внедрения: объект реп. создается для к. http-запроса

**Global.asax:**файл, есть во всех прил ASP.NET, нужен чтоб обраб. события ЖЦ приложения  
Applicaton\_Start() вызыв. при запуске прил  
с пом. DependencyResolver.SetResolver() регистр. сопоставитель зависимостей (NIConfig)

**Controller:**к-р() – несмотря на то, что в к-р не передается никакой конкр. реализации, все будет работать. Как тут внедр. зависимости:  
1) фреймворк MVC получ. запрос и обращ. к к-ру DictController  
2) фреймворк MVC обращ. к классу сопост. завис. (NIConfig)  
3) ninject смотрит на к-р DictController, видит что там исп. зависимость то ин-а IPhone, для к-го он устан. сопоставление с конкр. реализацией

**Helpers:**

\* разраб. внешний **html-helper**, кот. реализует формы для ввода и корректировки строки справочника

позволяющие генерировать html-код

**Views/Partial:**

\* разраб. partial-предст, позвол. отобразить 1 строку справочника  
исп. это предст в цикле вывода списка телефонов

**Views/\_Layout:**

5) \* обесп. работу общего макета для всех представлений прил.  
макет сод. общую статич. часть и секции: для вывода заголовка, осн. части, подвала html-страниц

в \_ViewStart.cshtml – указ. Layout наш

**ВОПРОСЫ:**

1) Поясните понятие «внедрение зависимости» DI.

Прогр. мех, позв. авто- создавать прогр. объект с заданным ЖЦ (когда будет создаваться, разрушаться и как он будет внедряться (непосредственно или ч/параметры методов) – мы это м. определить)

2) Для чего применяется DI  
\* чтобы создавать слабосвязанный код  
\* для повторного применения кода  
Появ независимость от конкр реализации объекта, кот реализует ин-с  
чтоб мб репозиторий (объект модели) подменить и потом просто перепишем конфигуратор, свяжем например с классом А2

3) Что такое Ninject?  
есть неск. . IoC-контейнеров (фреймворков), позв. использ. DI: NinJect, AutoFac, Unity…  
(это наз. IoC – инверсия управления)

4) Поясните принцип подключения и конфигурации Ninject в ASP.NET-приложении  
надо скачать неск. библ из Nuget (в лк)

5) Каким образом мб выполнено DI в ASP.NET-приложении

**\* constructor injection**: все зависимости передаются через конструктор класса.

**\* setter injection**: разработчик добавляет setter-метод, с помощью которого инжектор внедряет зависимость

**\* interface injection**: зависимость предоставляет инжектору метод, с помощью которого инжектор передаст зависимость. Разработчики должны реализовать интерфейс, предоставляющий setter-метод, который принимает зависимости

6) Назовите 4 вида DI, кот. позволяет сделать Ninject в ASP.NET-приложении и поясните их особенности

7) Поясните понятие «хэлпер»  
позволяющие генерировать html-код

8) Объясните последовательность разработки внутренних и внешних хэлперов

\* Встраиваемые хелперы объявляются с помощью дескриптора @helper в любом месте представления. Использоваться они могут только в том представлении, в котором объявлены

\* если хелпер очень объемный, то он может очень сильно захламлять разметку представления. И в этом случае его лучше вынести в отдельный файл в виде методов расширений класса HtmlHelper.

9) Перечислите стандартные хэлперы входящие в состав ASP.NET

\* linkText – лэйбл для ссылки  
\* actionName  
\* routeValues – массив значений, привод. к методу  
\* controllerName  
\* htmlAttributes – набор html-атрибутов  
\* protocol (https)  
\* hostname  
\* fragment