CodeTalk5 – Ninject

namespace CodeTalk.Controllers

{

    public class HomeController : Controller

    {

        //5a

        //Constructor

        //private  TalkService talkService;

          private ITalkService talkService;

        //private  CommentService commentService;

          private ICommentService commentService;

        //public HomeController()

          public HomeController(ITalkService \_talkService, ICommentService \_commentService)

        {

            //  talkService = new TalkService();

                talkService = \_talkService;

            //  commentService = new CommentService();

                commentService = \_commentService;

        }

        //private TalkService talkService;

        //private CommentService commentService;

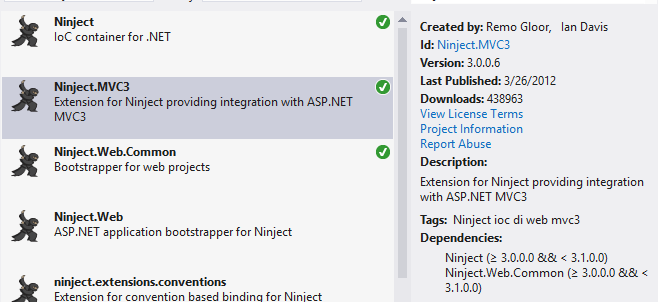
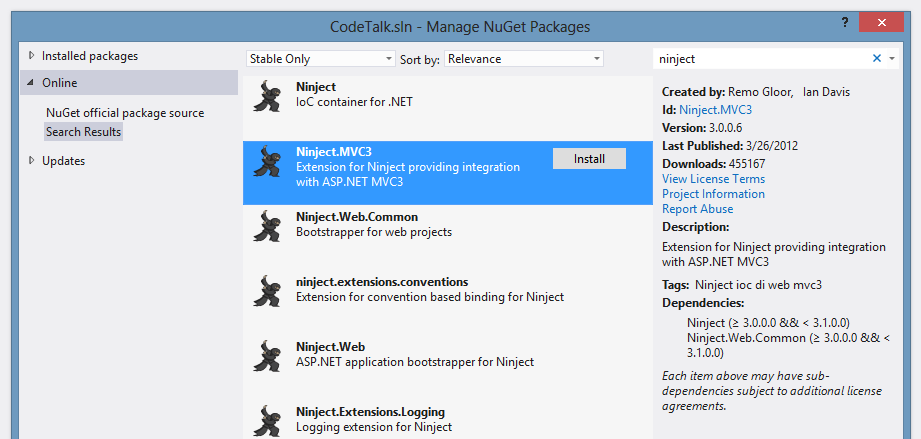
        //public HomeController()

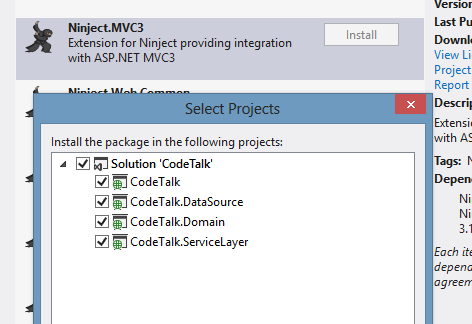
        //{

        //    talkService = new TalkService();

        //    commentService = new CommentService();

        //}





[assembly: WebActivator.PreApplicationStartMethod(typeof(CodeTalk.App\_Start.NinjectWebCommon), "Start")]

[assembly: WebActivator.ApplicationShutdownMethodAttribute(typeof(CodeTalk.App\_Start.NinjectWebCommon), "Stop")]

namespace CodeTalk.App\_Start

{

    using System;

    using System.Web;

    using Microsoft.Web.Infrastructure.DynamicModuleHelper;

    using Ninject;

    using Ninject.Web.Common;

    //5b.1

    using CodeTalk.Domain.Contracts.Services;

    using CodeTalk.ServiceLayer;

    using CodeTalk.Domain.Contracts.Repositories;

    using CodeTalk.DataSource;

    using CodeTalk.DataSource.Repositories;

    public static class NinjectWebCommon

    {

        private static readonly Bootstrapper bootstrapper = new Bootstrapper();

        /// <summary>

        /// Starts the application

        /// </summary>

        public static void Start()

        {

            DynamicModuleUtility.RegisterModule(typeof(OnePerRequestHttpModule));

            DynamicModuleUtility.RegisterModule(typeof(NinjectHttpModule));

            bootstrapper.Initialize(CreateKernel);

        }

        /// <summary>

        /// Stops the application.

        /// </summary>

        public static void Stop()

        {

            bootstrapper.ShutDown();

        }

        /// <summary>

        /// Creates the kernel that will manage your application.

        /// </summary>

        /// <returns>The created kernel.</returns>

        private static IKernel CreateKernel()

        {

            var kernel = new StandardKernel();

            kernel.Bind<Func<IKernel>>().ToMethod(ctx => () => new Bootstrapper().Kernel);

            kernel.Bind<IHttpModule>().To<HttpApplicationInitializationHttpModule>();

            RegisterServices(kernel);

            return kernel;

        }

        /// <summary>

        /// Load your modules or register your services here!

        /// </summary>

        /// <param name="kernel">The kernel.</param>

        private static void RegisterServices(IKernel kernel)

        {

            //5b.0

            kernel.Bind<ITalkService>().To<TalkService>().InRequestScope();

            kernel.Bind<ICommentService>().To<CommentService>().InRequestScope();

            //5e,5f

            kernel.Bind<ITalkRepository>().To<TalkRepository>().InRequestScope();

            kernel.Bind<ICommentRepository>().To<CommentRepository>().InRequestScope();

            //5g,5h

            kernel.Bind<CodeTalkContext>().To<CodeTalkContext>().InRequestScope();

            //5i

            //kernel.Bind<CodeTalkContext>().To<FakeCodeTalkContext>().InRequestScope();

        }

    }

}

namespace CodeTalk.DataSource.Repositories

{

   public class CommentRepository: ICommentRepository

   {

//5h

      // Constructor

      private CodeTalkContext codeTalkContext;

      public CommentRepository(CodeTalkContext \_codeTalkContext)

      {

          codeTalkContext = \_codeTalkContext;

      }

      //// Constructor

      //private CodeTalkContext codeTalkContext;

      //public CommentRepository()

      //{

      //    codeTalkContext = new CodeTalkContext();

      //}

namespace CodeTalk.DataSource.Repositories

{

 public class TalkRepository: ITalkRepository

 {

   //5g

   //Constructor

   private CodeTalkContext codeTalkContext;

   public TalkRepository(CodeTalkContext \_codeTalkContext)

   {

       codeTalkContext = \_codeTalkContext;

   }

   ////Constructor

   //private CodeTalkContext codeTalkContext;

   //public TalkRepository()

   //{

   //    codeTalkContext = new CodeTalkContext();

   //}

namespace CodeTalk.ServiceLayer

{

    public class TalkService : ITalkService

    {

      //5e

     //Constructor

     private ITalkRepository talkRepository;

     public TalkService(ITalkRepository \_talkRepository)

     {

         talkRepository = \_talkRepository;

     }

     ////Constructor

     //private ITalkRepository talkRepository;

     //public TalkService()

     //{

     //    talkRepository = new TalkRepository();

     //}

namespace CodeTalk.ServiceLayer

{

   public class CommentService : ICommentService

   {

    //5f

    //Constructor

    private ICommentRepository commentRepository;

    public CommentService(ICommentRepository \_commentRepository)

    {

        commentRepository = \_commentRepository;

    }

    ////Constructor

    //private ICommentRepository commentRepository;

    //public CommentService()

    //{

    //    commentRepository = new CommentRepository();

    //}

using CodeTalk.Domain.Contracts.Services; //5d

namespace CodeTalk.Controllers

{

    public class CommentController : Controller

    {

        //5d

        //Constructor

        private ICommentService commentService;

        public CommentController(ICommentService \_commentService)

        {

            commentService = \_commentService;

        }

        //Constructor

        //private CommentService commentService;

        //public CommentController()

        //{

        //    commentService = new CommentService();

        //}

using CodeTalk.Domain.Contracts.Services; //5c

namespace CodeTalk.Controllers

{

    public class TalkController : Controller

    {

        //5c

        // Constructor

        private ITalkService talkService;

        public TalkController(ITalkService \_talkService)

        {

            talkService = \_talkService;

        }

        //// Constructor

        //private TalkService talkService;

        //public TalkController()

        //{

        //    talkService = new TalkService();

        //}

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using CodeTalk.Domain.Models;

using CodeTalk.Domain.Contracts.Repositories;

namespace CodeTalk.DataSource

{

    public class FakeCodeTalkContext:  CodeTalkContext

, ITalkRepository, ICommentRepository

    {

        //5i

        public DbSet<Talk> Talks { get; set; }

        public DbSet<Comment> Comments { get; set; }

        public List<Talk> aTalks { get; set; }

        public List<Comment> aComments { get; set; }

        public FakeCodeTalkContext()

        {

            // Initialize lists

            // Talks

            aTalks = new List<Talk>();

            Talk t = new Talk();

            t.Id = 1;

            t.Title = "Title1";

            t.Description = "Description1";

            t.Comments = new List<Comment>();

            t.DateCreated = Convert.ToDateTime("1/1/2013");

            t.DateModified = Convert.ToDateTime("1/1/2013");

            aTalks.Add(t);

            t = new Talk();

            t.Id = 2;

            t.Title = "Title2";

            t.Description = "Description2";

            t.Comments = new List<Comment>();

            t.DateCreated = Convert.ToDateTime("2/2/2013");

            t.DateModified = Convert.ToDateTime("2/2/2013");

            aTalks.Add(t);

            t = new Talk();

            t.Id = 3;

            t.Title = "Title3";

            t.Description = "Description3";

            t.Comments = new List<Comment>();

            t.DateCreated = Convert.ToDateTime("3/3/2013");

            t.DateModified = Convert.ToDateTime("3/3/2013");

            aTalks.Add(t);

            // Comments

            aComments = new List<Comment>();

            Comment c = new Comment();

            c.Id = 1;

            c.TalkId = 1;

            c.Body = "Body1.1";

            c.Commentor = "Commentor1.1";

            c.DateCreated = Convert.ToDateTime("1/1/2013");

            c.DateModified = Convert.ToDateTime("1/1/2013");

            aComments.Add(c);

            c = new Comment();

            c.Id = 2;

            c.TalkId = 1;

            c.Body = "Body1.2";

            c.Commentor = "Commentor1.2";

            c.DateCreated = Convert.ToDateTime("1/2/2013");

            c.DateModified = Convert.ToDateTime("1/2/2013");

            aComments.Add(c);

            c = new Comment();

            c.Id = 3;

            c.TalkId = 2;

            c.Body = "Body2.1";

            c.Commentor = "Commentor2.1";

            c.DateCreated = Convert.ToDateTime("2/1/2013");

            c.DateModified = Convert.ToDateTime("2/1/2013");

            aComments.Add(c);

            c = new Comment();

            c.Id = 4;

            c.TalkId = 2;

            c.Body = "Body2.2";

            c.Commentor = "Commentor2.2";

            c.DateCreated = Convert.ToDateTime("2/2/2013");

            c.DateModified = Convert.ToDateTime("2/2/2013");

            aComments.Add(c);

            c = new Comment();

            c.Id = 5;

            c.TalkId = 3;

            c.Body = "Body3.1";

            c.Commentor = "Commentor3.1";

            c.DateCreated = Convert.ToDateTime("3/1/2013");

            c.DateModified = Convert.ToDateTime("3/1/2013");

            aComments.Add(c);

            c = new Comment();

            c.Id = 6;

            c.TalkId = 3;

            c.Body = "Body3.2";

            c.Commentor = "Commentor3.2";

            c.DateCreated = Convert.ToDateTime("3/2/2013");

            c.DateModified = Convert.ToDateTime("3/2/2013");

            aComments.Add(c);

        }

        //Talk Repository Methods

        public IQueryable<Talk> GetTalks()

        {

            return aTalks.AsQueryable<Talk>();

        }

        public Talk GetTalkById(int talkId)

        {

            throw new NotImplementedException();

        }

        public bool AddTalk(Talk newTalk)

        {

            throw new NotImplementedException();

        }

        public void EditTalk(Talk updatedTalk)

        {

            throw new NotImplementedException();

        }

        public void DeleteTalk(int deleteTalkWithThisId)

        {

            throw new NotImplementedException();

        }

        //Comment Reposistory Methods

        public IQueryable<Comment> GetComments()

        {

            throw new NotImplementedException();

        }

        public Comment GetCommentById(int commentId)

        {

            throw new NotImplementedException();

        }

        public bool AddComment(Comment newComment)

        {

            throw new NotImplementedException();

        }

        public void EditComment(Comment updatedComment)

        {

            throw new NotImplementedException();

        }

        public void DeleteComment(int deleteCommentWithThisId)

        {

            throw new NotImplementedException();

        }

    }

}