New Project > ASP.NET MVC4 “CodeTalk” > Internet Application

Rename CodeTalk project to CodeTalk.**Web**

Add 3 Class Library projects:

* CodeTalk.DataSource

DataSource

* CodeTalk.Domain
* CodeTalk.ServiceLayer

Add references:

CodeTalk.**DataSource** ->

* CodeTalk.Domain
* EntityFramework

CodeTalk.**Domain** (none)

ServiceLayer

Domain

CodeTalk.**ServiceLayer**

* CodeTalk.DataSource
* CodeTalk.Domain

CodeTalk.**Web**

* CodeTalk.Domain
* CodeTalk.ServiceLayer
* EntityFramework

**Solution Structure**

Web

4 projects:

|  |  |
| --- | --- |
| CodeTalk.Domain | \Models  \Contracts\Repositories  \Contracts\Services |
| CodeTalk.DataSource | \  \Repositories |
| CodeTalk.ServiceLayer | \ |
| CodeTalk.Web | \Controllers  \Views |

**Roadmap**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1. | [Model](#Model) |  |  |  |  |
|  |  | CodeTalk.**Domain** | Models\ | Talk.cs |  |
|  |  |  |  |  |  |
| 2. | Interfaces |  |  |  |  |
|  |  | CodeTalk.**Domain** | Contracts\Repositories\ | ITalkRepository.cs | using CodeTalk.Domain.Models; |
|  |  |  |  |  |  |
| 3. | Context |  |  |  |  |
|  |  | CodeTalk.**DataSource** | \ | CodeTalkContext.cs | using CodeTalk.Domain.Models; |
| 4. | Repository |  |  |  |  |
|  |  | CodeTalk.**DataSource** | \Repositories\ | TalkRepository.cs | using CodeTalk.Domain.Models;  using CodeTalk.Domain.Contracts;  using CodeTalk.Domain.Contracts.Repositories; |
|  |  |  |  |  |  |
| 5. | ServiceLayer |  |  |  |  |
|  |  | CodeTalk.**Domain** | Contracts\Services\ | ITalkService.cs | using CodeTalk.Domain.Models; |
|  |  |  |  |  |  |
|  |  | CodeTalk.**ServiceLayer** | \ | TalkService.cs | using CodeTalk.Domain.Models;  using CodeTalk.Domain.Contracts.Services;  using CodeTalk.DataSource;  using CodeTalk.DataSource.Repositories; |
|  |  |  |  |  |  |
| 6. | Controller |  |  |  |  |
|  |  | CodeTalk.**Web** | \Controllers | TalkController.cs | using CodeTalk.Domain.Models;  using CodeTalk.ServiceLayer; |
|  |  |  |  |  |  |
| 7. | Views |  |  |  |  |
|  |  | CodeTalk.**Web** | \Views\Shared | \_Layout.cshtml |  |
|  |  | CodeTalk.**Web** | \Views\Talk | Index.cshtml  Insert.cshtml | @model IEnumerable<CodeTalk.Domain.Models.Talk>  @model CodeTalk.Domain.Models.Talk |

**1.**

**Model**

CodeTalks.Model

**POCO Models**

no project references

domain objects

vehicles for data

properties that define data

EF will populate them

POCO

* Plain Old Class Object
* Independent and stand alone objects
* Easy to pass around

Web API > Unit of Work > Repositories > EF > db

 1.

namespace CodeTalks.Model

{

  public class Talk

  {

    public int Id { get; set; }

    public string Title { get; set; }

    public string Body { get; set; }

    public DateTime Created { get; set; }

    public bool Flagged { get; set; }

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

  }

}

namespace CodeTalks.Model

{

  public class Comment

  {

    public int Id { get; set; }

    public string Body { get; set; }

    public DateTime Created { get; set; }

    public int TalkId { get; set; }

  }

}

**2.**

**Interfaces**

CodeTalks.Data.Contracts

References:

* CodeTalks.Model

Just methods we need to get at data

//paging, ordering, grouping, sorting, filtering…

so IQueryable (rather than IEnumerable)

Just start with   
-GetTalks()   
-GetCommentsByTalk

Allows us to mock up a repo for testing later

namespace CodeTalks.Data.Contracts

{

  public interface ICodeTalksRepository

  {

    IQueryable<Talk> GetTalks();

    IQueryable<Talk> GetTalksIncludingComments();

IQueryable<Comment> GetCommentsByTalk(int talkId);

    bool Save();

    bool AddTalk(Talk newTalk);

    bool AddComment(Comment newComment);

  }

}

3.

Context – communication between code and db – conduit for crud

Entity Framework - Nuget package or Add reference

"DefaultConnection"- from security video?

namespace CodeTalks.Data

{

    public class CodeTalksContext: DbContext

    {

        public CodeTalksContext()

            : base("DefaultConnection")

        {

        }

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

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

    }

}

**4.**

**Repository** and **Unit of Work**

CodeTalks**.Data**

References:

* CodeTalks.Data.Contracts
* CodeTalks.Model
* EntityFramework

namespace CodeTalks.Data

{

  public class CodeTalksRepository : ICodeTalksRepository

  {

    CodeTalksContext \_ctx;

    public MessageBoardRepository(CodeTalksContext ctx)

    {

      \_ctx = ctx;

    }

    public IQueryable<Talk> GetTalks()

    {

      return \_ctx.Talks;

    }

    public IQueryable<Comment> GetCommentsByTalk(int talkId)

    {

      return \_ctx.Comments.Where(c => c.TalkId == talkId);

    }

    public bool Save()

    {

      try

      {

        return \_ctx.SaveChanges() > 0;

      }

      catch (Exception ex)

      {

        // TODO log this error

        return false;

      }

    }

    public bool AddTalk(Talk newTalk)

    {

      try

      {

        \_ctx.Talks.Add(newTalk);

        return true;

      }

      catch (Exception ex)

      {

        // TODO log this error

        return false;

      }

    }

    public IQueryable<Topic> GetTalksIncludingComments()

    {

      return \_ctx.Talks.Include("Comments");

    }

    public bool AddComment(Comment newComment)

    {

      try

      {

        \_ctx.Comments.Add(newComment);

        return true;

      }

      catch (Exception ex)

      {

        // TODO log this error

        return false;

      }

    }

  }

}

Version 2.0 – add in Comments

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1. | Model |  |  |  |  |
|  |  | CodeTalk.**Domain** | Models\ | Talk.cs |  |
|  |  | CodeTalk.**Domain** | Models\ | Comment.cs |  |
| 2. | Interfaces |  |  |  |  |
|  |  | CodeTalk.**Domain** | Contracts\Repositories\ | ITalkRepository.cs |  |
|  |  | CodeTalk.**Domain** | Contracts\Repositories\ | ICommentRepository.cs |  |
| 3. | Context |  |  |  |  |
|  |  | CodeTalk.**DataSource** | \ | CodeTalkContext.cs |  |
| 4. | Repository |  |  |  |  |
|  |  | CodeTalk.**DataSource** | \ | CodeTalkRepository.cs |  |
|  |  | CodeTalk.**DataSource** | \Repositories\ | TalkRepository.cs |  |
|  |  | CodeTalk.**DataSource** | \Repositories\ | CommentRepository.cs |  |
| 5. | ServiceLayer |  |  |  |  |
|  |  | CodeTalk.**Domain** | Contracts\Services\ | ITalkService.cs |  |
|  |  | CodeTalk.**Domain** | Contracts\Services\ | ICommentService.cs |  |
|  |  | CodeTalk.**ServiceLayer** | \ | TalkService.cs |  |
|  |  | CodeTalk.**ServiceLayer** | \ | CommentService.cs |  |
| 6. | Controller |  |  |  |  |
|  |  | CodeTalk.**Web** | \Controllers | TalkController.cs |  |
|  |  | CodeTalk.**Web** | \Contollers | CommentController.cs |  |
| 7. | Views |  |  |  |  |
|  |  | CodeTalk.**Web** | \Views\Shared | \_Layout.cshtml |  |
|  |  | CodeTalk.**Web** | \Views\Talk | Index.cshtml  Insert.cshtml |  |
|  |  | CodeTalk.**Web** | \Views\Comment | Insert.cshtml |  |

namespace CodeTalks.Data

{

  public class CodeTalksContext : DbContext

  {

    public CodeTalksContext()

      : base("DefaultConnection")

    {

      this.Configuration.LazyLoadingEnabled = false;

      this.Configuration.ProxyCreationEnabled = false;

   }

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

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

  }

}

Ninject

    /// <summary>

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

    /// </summary>

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

    private static void RegisterServices(IKernel kernel)

    {

#if DEBUG

      kernel.Bind<IMailService>().To<MockMailService>().InRequestScope();

#else

      kernel.Bind<IMailService>().To<MailService>().InRequestScope();

#endif

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

      kernel.Bind<ICodeTalksRepository>().To<CodeTalksRepository>().InRequestScope();

    }

5. ServiceLayer

5a. ITalkService

5b. TalkService

4.

//right click ICodeTalksRepository to implement

namespace CodeTalks.Data

{

    public class CodeTalksRepository: ICodeTalksRepository

    {

        public IQueryable<Talk> GetTalks()

        {

            throw new NotImplementedException();

        }

        public IQueryable<Comment> GetCommentsByTalk(int talkId)

        {

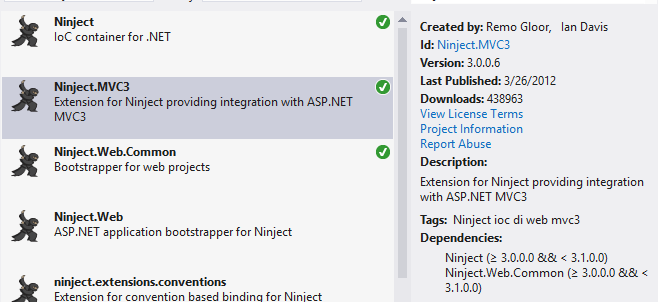
            throw new NotImplementedException();

        }

    }

}

4.1 Ninject



    /// <summary>

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

    /// </summary>

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

    private static void RegisterServices(IKernel kernel)

    {

#if DEBUG

      kernel.Bind<IMailService>().To<MockMailService>().InRequestScope();

#else

      kernel.Bind<IMailService>().To<MailService>().InRequestScope();

#endif

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

      kernel.Bind<ICodeTalksRepository>().To<CodeTalksRepository>().InRequestScope();

    }

4.2

//next… (a)

// works, but implies a new MessageBoardContext each time call GetTopics

// create and dispose

public class MessageBoardRepository : IMessageBoardRepository

{

public IQueryable<Topic> GetTopics()

{

var ctx = new MessageBoardContext();

return ctx.Topics;

// and it's disposable so…

**using (**var ctx = new MessageBoardContext()**)**

**{**

return ctx.Topics;

**}**

}

public IQueryable<Reply> GetRepliesByTopic(int topicId)

{

}

}

//next… (b)

// so create a single instance of ctx in the ctor

public class MessageBoardRepository : IMessageBoardRepository

{

**MessageBoardContext \_ctx;**

**public MessageBoardRepository()**

**{**

**\_ctx = new MessageBoardContext();**

**}**

public IQueryable<Topic> GetTopics()

{

return \_ctx.Topics;

}

public IQueryable<Reply> GetRepliesByTopic(int topicId)

{

}

}

//next… (c)

// but we'd prefer to have the ctx passed to us so we can use any one that already exists

public class MessageBoardRepository : IMessageBoardRepository

{

MessageBoardContext \_ctx;

public MessageBoardRepository(**MessageBoardContext ctx**)

{

return \_ctx = **ctx**;

}

public IQueryable<Topic> GetTopics()

{

}

public IQueryable<Reply> GetRepliesByTopic(int topicId)

{

return \_ctx.Replies.Where(r => r.TopicId == topicId);

}

}

PM> enable-migrations

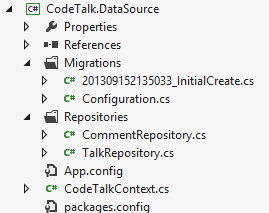
Checking if the context targets an existing database...

Detected database created with a database initializer. Scaffolded migration '201309152135033\_InitialCreate' corresponding to existing database. To use an automatic migration instead, delete the Migrations folder and re-run Enable-Migrations specifying the -EnableAutomaticMigrations parameter.

Code First Migrations enabled for project CodeTalk.DataSource.

This command has added a **Migrations** folder to our project, this new folder contains two files:

* **The Configuration class.** This class allows you to configure how Migrations behaves for your context. For this walkthrough we will just use the default configuration.  
  *Because there is just a single Code First context in your project, Enable-Migrations has automatically filled in the context type this configuration applies to.*
* **An InitialCreate migration**. This migration was generated because we already had Code First create a database for us, before we enabled migrations. The code in this scaffolded migration represents the objects that have already been created in the database. In our case that is the **Blog** table with a **BlogId** and **Name** columns. The filename includes a timestamp to help with ordering.



* **Add-Migration** will scaffold the next migration based on changes you have made to your model since the last migration was created
* **Update-Database** will apply any pending migrations to the database

PM> add-migration AddCommentClass

Scaffolding migration 'AddCommentClass'.

The Designer Code for this migration file includes a snapshot of your current Code First model. This snapshot is used to calculate the changes to your model when you scaffold the next migration. If you make additional changes to your model that you want to include in this migration, then you can re-scaffold it by running 'Add-Migration 201310072032267\_AddCommentClass' again.

namespace CodeTalk.DataSource.Migrations

{

    using System;

    using System.Data.Entity.Migrations;

    public partial class AddCommentClass : DbMigration

    {

        public override void Up()

        {

            CreateTable(

                "dbo.Comments",

                c => new

                    {

                        Id = c.Int(nullable: false, identity: true),

                        Body = c.String(),

                        TalkId = c.Int(nullable: false),

                        DateCreated = c.DateTime(nullable: false),

                        DateModified = c.DateTime(nullable: false),

                    })

                .PrimaryKey(t => t.Id);

        }

        public override void Down()

        {

            DropTable("dbo.Comments");

        }

    }

}

namespace CodeTalk.DataSource.Migrations

{

    using System;

    using System.Data.Entity.Migrations;

    public partial class InitialCreate : DbMigration

    {

        public override void Up()

        {

            CreateTable(

                "dbo.Talks",

                c => new

                    {

                        Id = c.Int(nullable: false, identity: true),

                        Title = c.String(),

                        Description = c.String(),

                        DateCreated = c.DateTime(nullable: false),

                        DateModified = c.DateTime(nullable: false),

                    })

                .PrimaryKey(t => t.Id);

        }

        public override void Down()

        {

            DropTable("dbo.Talks");

        }

    }

}

* Run the **Update-Database** command in Package Manager Console
* Code First Migrations will compare the migrations in our **Migrations** folder with the ones that have been applied to the database. It will see that the **AddBlogUrl** migration needs to be applied, and run it.

PM> update-database

Specify the '-Verbose' flag to view the SQL statements being applied to the target database.

Applying code-based migrations: [201310072032267\_AddCommentClass].

Applying code-based migration: 201310072032267\_AddCommentClass.

Running Seed method.

---