Demo project

From

Stephen Webb

Contents

[Introduction 1](#_Toc360342343)

[Project structure 2](#_Toc360342344)

[Solution and tests 3](#_Toc360342345)

[Database connection 3](#_Toc360342346)

[Tests 3](#_Toc360342347)

[Data Transfer Objects 3](#_Toc360342348)

[WCF Layer. 4](#_Toc360342349)

[Application Layer. 4](#_Toc360342350)

[Domain layer 4](#_Toc360342351)

[Database 4](#_Toc360342352)

## Introduction

The six required methods have been implemented.

Have used Entity Framework code first.

Have used Structure Map.

Logical layers implemented.

Have used Generic repository

Designed using TDD.

Implemented integration tests on the Application layer.

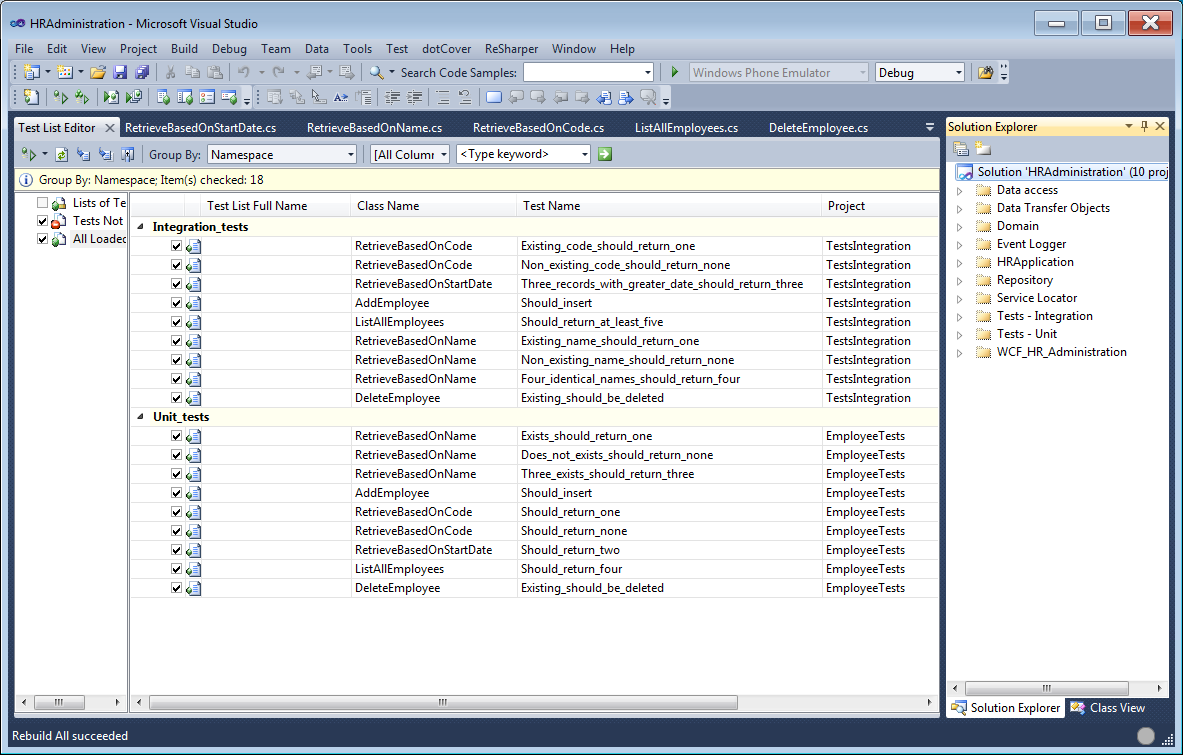
Database is created and seeded when the integration tests are run.

## 

## Project structure



## Solution and tests



## Database connection

If you want to run the integration tests and the WCF service then edit connection string in the App.config file in the following project: TestsIntegration

<connectionStrings>

<add name="DelarueSystemContext"

connectionString="

Data Source=SMALLBOY\SQLEXPRESS;Initial Catalog=Delarue\_HR\_System;

Integrated Security=True;

MultipleActiveResultSets=True"

providerName="System.Data.SqlClient" />

</connectionStrings>

The same connection string also exists in the App.config file of the WCF\_HR\_Administration project.

## Tests

The method relating to after date, will not return records with that data, only records after that date.

## Data Transfer Objects

I believe that internal classes should not be exposed via WCF methods. Therefore internal classes are transformed to external versions for public consumption.

## WCF Layer.

This is a very ‘thin’ layer and is a way of exposing the application to BizTalk.

## Application Layer.

In a more complex application, this is where unit of work is implemented. We coup use Aspect Oriented Programming to implement cross cutting concerns such as logging, security etc.

## Domain layer

The domain layer contains business logic.

The domain layer does not have try/catch. I believe that you should only have catches where you can do some sort of recovery from the error. So, exception handling is done in the layer above which is the application layer.

Also the domain layer does not contain this such as context.SaveChanges. Domain objects should not know anything about how to save or load themselves.

## Database

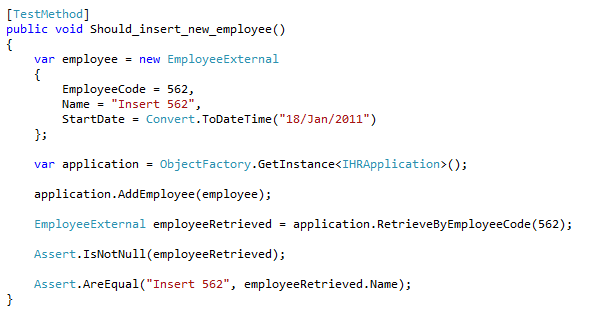
The database is created and seeded only if the integration tests are run.

The name of the created database is: Delarue\_HR\_System

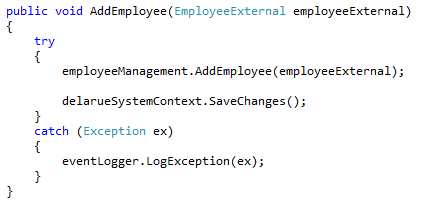
## Vertical slice of functionality.

This shows the code from an integration test down to the repository layer.

**LAYER: Tests - Integration**



**LAYER: Application\HRApplication**



**LAYER: Domain\EmployeeManagement**

