Contents

KinderSurprise

Date of Create: 28.01.2011
Date of last Update: 08.02.2011
Author: Enrico Gallus

Supposed to be a web-based c# application used to save and organize a kindersurprise collection.

0.1 Aim

- web based application
- using database to save information
- platform independent
- user-specific acces to the data
- provide a simple system to show information of categories, series and figures
- make every item editable, deletable and addable
- every serie and figure should consist of a store
- support images for series and figures
- import/export functionality for all the data
- has ebay-interface to sell figures/series
- offers all functions of a customer-based trading center

0.2 Developing Tools

I used to work under Debian Testing (Squeeze), using the following Repository for Mono: "deb http://debian.meebey.net/pkg-mono /"

- MonoDevelop v2.4
- Mono-Framework
- NHibernate v2.1.2.4000
- FluentNHibernate v1.1
- MySql v5.1
- NUnit v2.4.7
- Moq v4.0.10827

- NhLambdaExtensions v1.0.10.0
- structuremap v2.6.1
- rabbitvcs-nautilus

0.3 Patterns

- Model-View-Presenter
- Dependency Injection
- N-Tier
- Test-Driven-Development
- UnitOfWork

0.4 Installation

0.4.1 Preparation

Install the compontents mentioned in the Section D̈eveloping Tools:

0.4.2 VCS

Get the Source Code from the project via Berlios. I recommend the rabbitvcs-nautilus package from the distribution. Handling vcs can't be easier and you can handle everything over nautilus.

0.4.3 Setup Database

The first step is to set up the database and testdatabase with MySql. Please create an user for your mysql instance and create a database named "kindersurprise" and "testkindersurprise". Afterwards open the projectdir and change to the script folder. In there you will find all scripts necessary for the database structure. The name is organised as date followed by the function. Please execute the containing scripts for both databases in chronological order. If no errors occured and the structure was created on both databases, than please also execute the testscripts in the test database.

0.4.4 Setup Database Connection

Please open the project with MonoDevelop. Use the sln-file in the projectDir to open the solution. You have to edit the WebTest.config file in the following projects

- KinderSurprise.DAL.Test
- KinderSurprise.MVP.Model.Test

- KinderSurprise.MVP.Presenter.Test

Customize the KinderSurpriseConnection String. Set the database name to "testkindersurprise" if you used the name for the testdatabase. Also change the ID to the user you have created in the step before. Change the password.

Also edit the Web.Config in the KinderSurprise.MVP.View subproject and make the same changes but instead of using the testdatabase use "kindersurprise".

0.4.5 Last Step

To verify if everything works well, please execute the unittests defined in the the three test projects. Compile the project and try to start. If any problem or error occure, please don't hesitate to contact me.

0.5 Using .net under Visual Studio

If you rather would use visual studio under windows than to use the mono framework, you have to change to modify some parts of the project. After open the solution file of the project the ui-project will maybe not be loaded correctly. If this error occured than you have to modify the solution file of the $KinderSurprise.MVP.View$ project like described in the file $windows_changes.txt$. Please comment out the part existing in the $windows_changes.txt$ in the solution file of the $KinderSurprise.MVP.View$. And copy the $windows_changes.txt$ content in the soltution file. Afterwards reload the project.
On the other side the $Nunit.framework.dll$ does not belong to the lib order. Mono is using the class directly from the system. So the references under visual studio are not correct. Delete all incorrect references to the dll in all the test projects. Copy the Nunit.Framework.dll to the lib directory and reference in the test-projects this libary.

0.6 Workflows

0.6.1 Working with the database

If you make changes to the database, please save all the commands to a script which can be added to the script-folder. So other developer has it as easy as you, to setup the project. No need to mention the filename-specification!?

0.6.2 Submitting of Code changes

If you want to commit any changes. Please be sure there are no broken unittests. I'm still looking for a codecoverage tool under linux which works with c#. If you have any proposal, so please let me know. If there are more developer, maybe we can think about using continous integration.

0.7 Structure and Workaround

0.7.1 View -The GUI

The view is the layer which provides the graphical user interface implemented with asp.net framework. The layer currently consists of one default page and an usercontrol for category, figur and serie. There exists also an account part, but it was automatically generated and should be reworked in the future. The default page contains a treeview on the left site to organize a view of all categories, series and figurs in a relation. If an element in the treeview was clicked the corresponding usercontrol should be open on the default page. Here you can work with the items, e.g. edit the properties. On the other site you sould be able to delete, add and move the item to another parent item.
Each page and usercontrol contains all properties (user control elements) as encapsulation. There is no business logic in this layer. All handling should be done by the presenter.
Featues:

- Think about an icon for the application
- The account
  - Register and Login should be implemented in c# and using less javascript (none if possible)
  - Think of a secure way to implement the login handling (password-encrypting-decrypting, ..)
  - Different Options should be activated if the user is logged in.
    - layout for the account, e.g. change personal information or password
    - The trading function should be activated
    - own message system to let user communicate with each other
- User notification if he wants to delete an item which has still childs.

0.7.2 The presenter

The interfaces defined in the presenter layer contain all the user controls of our gui. In this layer we can handle the changes of the layout of the user controls and use the model for the business behaviour.

0.7.3 The model

The model contains service methods for the repositories. And business code like the picturehandling, an validator and so on.

0.7.4 DAL - data access layer

The data access layer got on class for each table in the database. This is the lowest layer we can reach. NHibernate is used to handle the object - relational mapping between our application and the database in the background. The Code is in such a way encapsulated that changing the dbms does not require code changes. The repositories provide one method for one process (save, update, delete, getting by id or more complex queries.

0.7.5 The workaround

The user interact with our application. The view call the presenter and will tell the current state of the user control elements and so on. The presenter delegates the task to the model and the DAL to collect and edit the information. The dal is an interface for the database. The model is the workaround for our application.

0.7.6 Database model

The Layer Model contains the objects we are using in the application. This objects are used by the mapper.

0.7.7 Mapper

The mapper is using the FluentNHibernate framework to "connect" our objects with the corresponding relational tables in the database.

0.7.8 Bootstrap

Bootstrap contains our solution for the dependency injection pattern. There are a productive and a testing container.

0.7.9 NHBootstrap

This layer contains our implementation of the unit of work pattern.

0.7.10 Test

Some projects have an corresponding test project. Here is the testing of the behaviour of our units.