

Re-form: PhoneBook

Level 100 - Hands on Labs

This document is part of the re-motion Core Framework ([www.re-motion.org](http://www.re-motion.org)) Copyright (C) 2005-2011 rubicon informationstechnologie gmbh, www.rubicon.eu

The re-motion Core Framework is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

re-motion is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with re-motion; if not, see http://www.gnu.org/licenses.

Table of Contents

[1 Goals 1](#_Toc291680804)

[2 Lab 1: Using UIGEN 1](#_Toc291680805)

[2.1 Exercise: Understanding UI Generators 2](#_Toc291680806)

[2.1.1 Questions and Discussions 3](#_Toc291680807)

[2.2 Exercise: Using UI Gen 4](#_Toc291680808)

[2.2.1 Task 1: Preparing UIGEN 4](#_Toc291680809)

[2.2.2 Task 2: Using UIGEN 5](#_Toc291680810)

[2.2.3 Task 3: Testing UIGEN 5](#_Toc291680811)

[2.2.4 Questions and Discussions 6](#_Toc291680812)

[3 Lab 2: First steps with re-call 6](#_Toc291680813)

[3.1 Exercise: Understanding re-call 6](#_Toc291680814)

[3.1.1 Questions and Discussions 7](#_Toc291680815)

[3.2 Exercise 1: Get Rid of Exceptions 7](#_Toc291680816)

[3.2.1 Task 1: Get Rid of WxeExecuteNextStepException 7](#_Toc291680817)

[3.2.2 Questions and Discussions 8](#_Toc291680818)

[4 Lab 3: First steps with re-bind 8](#_Toc291680819)

[4.1 Exercise 1: Globalization 8](#_Toc291680820)

[4.1.1 Task 1: Fixing tab names 8](#_Toc291680821)

[4.1.2 Task 2: Adding Globalization on Properties 8](#_Toc291680822)

[4.1.3 Questions and Discussions 9](#_Toc291680823)

[4.2 Exercise 3: Customizing Content 9](#_Toc291680824)

[4.2.1 Task 1: Display names 9](#_Toc291680825)

[4.2.2 Task 2: Customizing a BOC List 10](#_Toc291680826)

[4.2.3 Questions and Discussions 11](#_Toc291680827)

[4.2.4 Task 3: Adding Icons 11](#_Toc291680828)

[4.2.5 Questions and Discussions 12](#_Toc291680829)

[4.2.6 Task 4: Adding LocationPicker to Person Page 15](#_Toc291680830)

[4.2.7 Questions and Discussions 17](#_Toc291680831)

[4.2.8 Task Add Delete 12](#_Toc291680832)

[4.2.9 Questions and Discussions 13](#_Toc291680833)

[4.2.10 Task Add Delete 13](#_Toc291680834)

[5 Final Questions 15](#_Toc291680835)

[6 Lab Summary 17](#_Toc291680836)

# Goals

Estimated time: 30 Minutes

Goal of this HOL is to teach the fundamentals of re-form, the presentation layer of re-motion. re-form contains re-bind, the data binding component with several controls, and re-call, the transactional navigation system.

In this lab you will continue to work on the phonebook sample, which you started with the HOL for re-store.

All exercises in this HOL can be solved step by step. Each exercise is designed to be finished with an executable result.

During this HOL you will encounter sections for questions and discussions. Answers to some questions might be easy. Other questions could be tricky. In case you are doing this exercise with other developers, we encourage you to talk about your ideas and your proposed solutions.

Please have also a look on the further readings section in the end of this HOL. They provide some suggestions on how you might want to proceed to learn more.

After this exercise you will be able to understand the advantages of re-form and to compare them with other technologies.

Topics:

|  |
| --- |
| * Lab1: Using UIGEN to generate a web application project from the domain * Lab2: First steps with re-call, the transactional navigation between forms * Lab3: First steps with re-bind, the binding between control and data |

**Requirements:**

Visual Studio 2010 must be installed on your working PC. Additionally, access to a SQL Server 2008 is required. You must have at least db\_owner rights.

Get the re-motion version from <https://www.re-motion.org/builds/>. This HOL was verified to be working with version 1.13.101.

This HOL continues where the re-store HOL has ended. If you have not done the re-store HOL yet, please finish this HOL or copy the files of a finished solution to C:\PhoneBook.

This HOL was tested with Internet Explorer 9 and Google Chrome 12.

# Lab 1 Building UIGEN

## Exercise: Building UIGEN

UIGEN.exe is part of the re-motion contribution project. UIGEN is shipped with a stable version of re-motion. If you are working with a development version, you might have to build it.

If you work with a production release, you can skip this step. To make sure what app you have, please check:

* Is a folder UIGenTemplates available in the remotion directory?
* Is under net-3.5\bin\debug a UIGen.exe file available?

If these artefacts are missing, you have downloaded a version without UIGEN. Therefore, you have to build UIGEN.

### Task 1: Building UIGEN

1. **Get latest source for UIGEN from**

<https://svn.re-motion.org/svn/Remotion-Contrib/Remotion.ObjectBinding.Web.CodeGenerator/trunk/>

If you have not worked with subversion before, we suggest that you install a subversion client such as TortoiseSVN to get the source code.

1. **Overwrite the lib directory with the re-motion binaries you want to use**
2. **Build the project**
3. **Copy the complied UIgen.\* files to your binaries for your PhoneBook Sample Application**
4. **Copy the folder Templates to your remotion folder for your sample app**
5. **Rename the directory Template to UIGenTemplates**

### Questions and Discussions

* You have learned that re-motion is divided into a core project and a contribution project. In the core project, there is all core functionality, the contribution projects contains additional, but not essential tools. What is your opinion about this strategy?

# Lab 2: Using UIGEN

## Exercise: Understanding UI Generators

WYSIWYG editors for GUIs are well known to developers. One of the first things you may have learned was to add text box controls and labels to create your first customized “Hello World” application.

Many new applications are created with a “controls into the designer first” principle in mind, which became popular with RAD development. .NET Developers create their Visual Studio GUI project, drag some controls from the Toolbox to the Designer and arrange them. Once the GUIs are finished, they add the corresponding logic by adding class libraries.

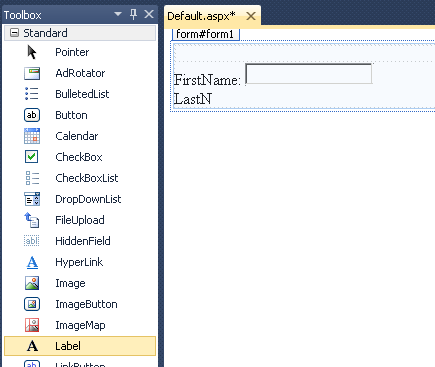


Illustration 1: Designing a Web GUI from the Scratch

Once you have a GUI, you are able to define what the application is doing. The next step would be to take care about how the application is processing the user input.

Working with re-motion is different. It can be defined in four steps.

1. **Create your Domain**
2. **Generate your Database Layer**
3. **Generate your Presentation Layer**
4. **Customize and Adapt**

Step 1 and 2 shall already be done in the HOL for re-store. The next question is how a web application layer can be generated from classes.

In simplified words: Every data type has its “default control”. Strings can be entered or changed in text boxes, date in date time pickers, enums in radio buttons, etc. Therefore, a class person with the properties string FirstName, string LastName, DateTime BirthDay would be two TextBoxes and one DateTimePicker with a Save and Cancel Button in a standard data input form.

So why do we need to draw the standard pages by designer, if we could generate search and edit forms for every entity. With the preconditions, it is possible to generate a web application based on the domain objects from a template. The generated forms contain an edit and search form for every domain object. Each property is represented by a control corresponding to its data type.

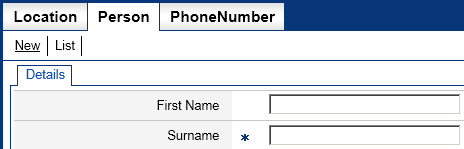


Illustration 2: Generated GUI for Person

The generated template can then be edited and adapted to individual needs.

### Questions and Discussions

* WYSIWYG editors can be troublesome. Many products have been known for the instability or the lack of performance of their designers. Do you know products or frameworks, where the designer is the weak spot? Why do you think the designer is the weak spot?
* Are there some additional common page controls that make sense to be generated for a form?
* Are there standard C# data types that cannot be mapped to standard controls?
* What are the advantages by generating web projects over designing them?
* Discuss the following statements about UIs. Some of them are definitely to be discussed or dangerous. Which of them do you consider as good which not?
* UIs must me consistent. It is a bad idea to solve a problem in a specific situation in a way and in similar situation in a different way.
* Every UI must support keyboard only input
* UI must be responsive. Every action must trigger an immediate response.
* If the UI is bad, the whole application is worthless, even if the business logic is perfect.
* Users love a well-designed GUI that looks good. If your UI does not look good, your app lacks quality (even though there are no functional problems).
* Modern UI controls with 3D effects are essential. The more new technologies, the better. They improve the general work performance and give the user a new feeling of working.
* You can neglect old browsers to make your GUI outstanding.
* It is hell, if you have to reenter data, because the data has been lost due to a session time out or something similar.
* UIs shall be designed for the power user. The intelligent user shall be enabled to work fast and he shall also be supported by some advanced complex controls. The lame shall learn or quit using the app.
* People are different and their perception and the way they work is different. So there is no good UI, but only UIs that meet the requirement of users.
* The best way to prevent problems with the UI is to prevent that a user enters wrong data.
* If the application does not work appropriate, it is the users fault! The best way of fixing this is to log every user action, pack the logs, mail them to the boss and instruct the boss how he can blame his employee for obvious wrong usage of the apps.
* The more information, the less focus. Some people say that google became dominant as they kept the design simple and focused whereas the competition had stuffed their site with too much information.
* Everyone loves to read. So it is essential to pack as much information as possible on a page. Even we have to repeat ourselves, the more information the better.
* Check out some additional thoughts about usability (see further links as samples). Discuss approaches such as:
* <http://www.smashingmagazine.com/2008/01/31/10-principles-of-effective-web-design/>
* <http://www.sylvantech.com/~talin/projects/ui_design.html>
* <http://www.ambysoft.com/essays/userInterfaceDesign.html>
* What do you think makes a UI successful? Add your own statements about a good UI!
* What is required to design a GUI to meet Web Accessibility features? Why do you think this important?
* One of the techniques to master in UI development is data binding. If you are new to this topic, research: What is data binding? How can this be achieved? How does a two way data binding differ from a one way data binding? What is an observer pattern?
* **Expert Question:** In which use cases would you recommend Microsoft Sharepoint, in which re-motion?

## Exercise: Using UI Gen

The logical steps to create the GUI of your DDD application are:

1. **Create your domain (which should already be done in our previous lab for re-store)**
2. **Prepare a code template to generate the GUI**
3. **Use UIGen.exe to generate your Visual Studio Web Project containing all forms from the command line**
4. **Adapt your generated UI to your own needs.**

### Task 1: Preparing UIGEN

UIGEN.EXE is a command line tool to generate a Visual Studio Web Project based on

* Compiled domain objects (from your domain project)
* A code template

To generate the UI for the phonebook application with UIGEN, you need to

1. **Verify that your domain project is compiled without any errors.**
2. **Add a PhoneBook.xml file to the directory C:\PhoneBook**
3. **Use the following content for PhoneBook.xml and adapt the directories to match your projects and template.**
4. **DON’T FORGET TO MATCH VARIABLE STRONG\_SUPPLEMENT TO YOUR WORKING RE-MOTION VERSION!**

<?xml version="1.0" encoding="utf-8" ?>

<!-- ATTENTION! Only ABSOLUTE PATHS are supported by uigen.exe -->

<!-- DO NOT USE RELATIVE PATHS OR YOUR APPLICATION WILL BREAK -->

<applicationGenerator template="\PhoneBook\Remotion\UIGenTemplates\TabbedEditor\TabbedEditor.xml">

<settings

templateRoot="\PhoneBook\Remotion\UIGenTemplates\TabbedEditor"

targetRoot="\PhoneBook"

projectNamespaceRoot="PhoneBook.Web"

domainNamespaceRoot="PhoneBook.Domain"

/>

<projectReplacements>

<replace from="$ReferencesDir$" to="References" />

<!-- Domain -->

<!-- Copy the DomainProjectGuid from your PhoneBook.Domain.csproj file -->

<replace from="$RemotionResDirectory$" to="\PhoneBook\Remotion\res" />

<replace from="$DomainProjectGuid$" to="{E715FCED-AD7C-4EB7-9E30-EDF67D904A48}" />

<replace from="$DomainProjectName$" to="PhoneBook.Domain" />

<replace from="$DomainProjectAssembly$" to="PhoneBook.Domain" />

<replace from="$RemotionAssembly$" to="\PhoneBook\Remotion\net-3.5\bin\Debug" />

<replace from="$WxeEngine$" to="\PhoneBook\Remotion\net-3.5\bin\Debug" />

<replace from="$WebClientName$" to="PhoneBook.Web" />

<replace from="$WebClientAssembly$" to="PhoneBook.Web" />

<replace from="$STRONG\_SUPPLEMENT$" to="Version=1.13.101.2, Culture=neutral,

PublicKeyToken=fee00910d6e5f53b" />

<!-- USER -->

<replace from="$USER\_APPNAME$" to="PhoneBook Web Sample" />

<replace from="$USER\_DEFAULT\_ASPX\_TOPIC$" to="This application was generated by UIGen.exe" />

<replace from="$USER\_DEFAULT\_ASPX\_STARTINFO$" to="To start the application click 'Start'" />

<replace from="$USER\_DEFAULT\_STARTPAGE$" to="EditLocation.wxe?WxeReturnToSelf=true" />

<replace from="$USER\_STATUSBAR$" to="yes" />

<replace from="$USER\_CLASSIC\_APPSTYLE$" to="false" />

<replace from="$USER\_STORAGEPROVIDER$" to="PhoneBookDB" />

<replace from="$USER\_CATALOGNAME$" to="PhoneBook" />

</projectReplacements>

</applicationGenerator>

### Task 2: Using UIGEN

1. **Go to the command line**
2. **Call** c:\PhoneBook\Remotion\net-3.5\bin\debug\UIGen.exe /uigen:C:\PhoneBook\PhoneBook.xml /asmdir:C:\PhoneBook\PhoneBook.Domain\bin\Debug

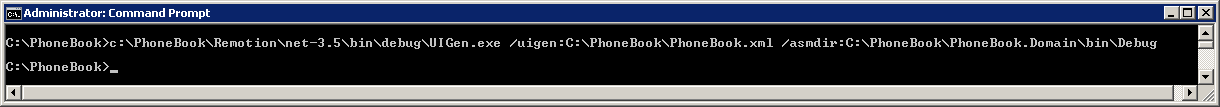


Illustration 3: UIGen on Console

### Task 3: Testing UIGEN

The UI Gen should have generated PhoneBook.Web Directory in C:\PhoneBook\

1. **Add the generated project to your solution in Visual Studio**
2. **Set the generated project as startup project**
3. **Build and run**

### Questions and Discussions

* Investigate the phonebook.xml file and try to find out what you can adapt!
* You can create a corporate design via templates to enable you and your fellow developers to create branded applications on a mouse click. On a scale from one to ten, how powerful is this feature?
* The default template is a tabbed editor: Forms are grouped by tabs. Of what alternative structures can you think of?
* What can you configure UIGenTemplate directory?
* **Expert Questions:** Could you also create a Silverlight or ASP.NET MVC project based on the domain with UIGen? What would be required to create a tool that creates a Silverlight project?

# Lab 2: First steps with re-call

## Exercise: Understanding re-call

In the first Lab you have learned how to generate a web project from the domain layer.

This generated web project contains Edit- and SearchForms for all domain objects. But before we cover the forms in detail, we have a look on the navigation between pages with re-form.

Traditionally, navigation is done the following way in a web application:

1. **user clicks control**
2. **control fires event**
3. **event-handler causes Server.Transfer or Response.Redirect to some specific page**

Depending on the implementation, the information in caller page is either lost or kept in a state. If you want to keep the state, you have to implement it.

The idea of re-call different! It is: *Treat a page call like a method call!*

Or in other words:

1. **user clicks control**
2. **control fires event**
3. **event-handler calls other page (with or without parameters)**
4. **On closing the called page, the called page can return a value**
5. **Once the page is closed, the caller page will be restored (probably the return value of the called page will be set)**



Illustration 4: Start Screen

### Questions and Discussions

* **Expert Question:** If you want to dig deeper it is essential to understand the concept of Web Handlers. Read about them and discuss what they do in groups!
* **Expert Question**: Talk about the idea: Calling pages like a method. What impact does it have?
* **Expert Question:** Discuss the topic session state management in web pages!
* **Expert Question:** Why can browser back, forward or refresh buttons lead to problems?

## Exercise 1: Get Rid of Exceptions

If you try to add a new entry during a debug session, you encounter the following error message after you have clicked on the save button:

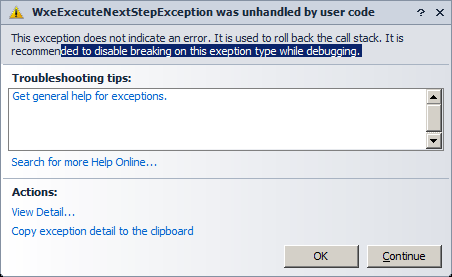


Illustration 5 WxeExecuteNextStepException

Due to the concept of ASP.NET, this is the only way to navigate between pages in the code. Therefore it is not an error. In the following task, we will get rid of this exception.

### Task 1: Get Rid of WxeExecuteNextStepException

To get rid of the exception warnings, you have to

1. **CTRL + ALT + E (Editing Exception)**
2. **Add Remotion.Web.ExecutionEngine.WxeExecuteNextStepException to the Common Language Runtime Expections**
3. **Uncheck User-unhandled**

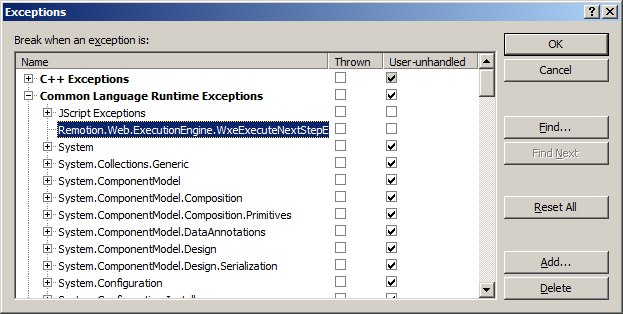


Illustration 6: Getting rid of WXE Exceptions

### Questions and Discussions

* **Expert Question:** This behavior to roll back a call stack is an “exception to the usage of exceptions.” Why is this required? Try to explain it with your own words.

# Lab 3: First steps with re-bind

re-motion comes with a rich set of controls with data binding support. They are covered in the component re-bind, which is a place holder for all controls including the data binding functionality.

## Exercise 1: Globalization

### Task 1: Fixing tab names

The tab names for the generated forms contain identifiers with ~ symbols. We can change this with the following steps

1. **Open global.resx file in PhoneBook.Web project**
2. **Change the names as you like them**

You can add more languages. You might want to try out to add globalized version specific languages.

### Task 2: Adding Globalization on Properties

After we have changed the tab names, we adapt identifiers in the controls like PhoneNumbers or FirstName. This has to be done in the domain projects as the domain projects stores the globalization information for its domain objects.

1. **Add class attributes to the class declaration of Person, PhoneNumber and Location in your domain project**

[MultiLingualResources("PhoneBook.Domain.Globalization.PhoneNumber")]

[MultiLingualResources("PhoneBook.Domain.Globalization.Person")]

[MultiLingualResources("PhoneBook.Domain.Globalization.Location")]

The Location class should look like

[DBTable]

[MultiLingualResources ("PhoneBook.Domain.Globalization.Location")]

public class Location : BindableDomainObject

Attention: If your domain is using the namespace PhoneBook.Domain, you will run into trouble. You can adapt either the project and assembly name or the string identifier in the class attribute.

1. **Add the class attribute**

[EnumDescriptionResource("PhoneBook.Domain.Globalization.Country")]

to enum Country

1. **Add a folder Globalization in the Domain Project**
2. **Add the files Location.resx, Person.resx, PhoneNumber.resx, Country.resx**
3. **Add at least the following properties to the resource files**

* Location.resx: Name property:ZipCode, Value: Zip
* Person.resx: Name property:FirstName, Value: First Name
* Person.resx: Name property:LastName, Value: Last Name

1. **Run application again and check what has changed**

### Questions and Discussions

* What are possible problems in general if you translate your page into another language?

## Exercise 3: Customizing Content

### Task 1: Display names

Looking on the form PhoneNumber, we get the last name of a person in the drop down list. This can be changed: We might want to get LastName, FirstName.

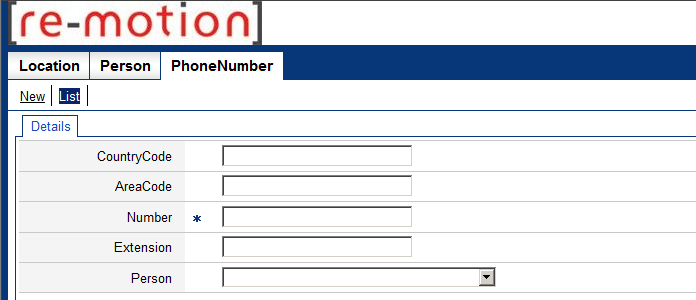


Illustration 7 EditPhoneNumberForm

1. **Replace DisplayName in class PhoneNumber with**

public override string DisplayName

{

get { return LastName; }

}

With

public override string DisplayName

{

get { return LastName + ", " + FirstName; }

}

Of course, it is possible to change the DisplayNames of the other classes too.

1. **Replace DisplayName in class Location with:**

public override string DisplayName

{

get

{

string country = Country == null ? string.Empty : " (" + EnumDescription.GetDescription(Country) + ")";

return Street + country;

}

}

1. **Replace DisplayName in class PhoneNumber with:**

public override string DisplayName

{

get

{

string displayName = "";

if (!String.IsNullOrEmpty (CountryCode))

{

displayName += CountryCode + " ";

}

if (!String.IsNullOrEmpty (AreaCode))

{

displayName += AreaCode + "/";

}

displayName += Number;

if (!String.IsNullOrEmpty (Extension))

{

displayName = displayName + "-" + Extension;

}

return displayName;

}

}

1. **Build and verify**

### Task 2: Customizing a BOC List

Looking at the search result of Location, we might get a list such as:

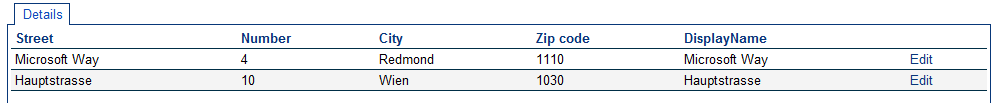


Illustration 8: Boclist

All properties are mapped by default to this list. In case of DisplayName this does not make sense.

In addition to this, we want to customize some fields. Let’s start with changing the SearchResultList of Location.

1. **Replace in SearchResultLocationControl**

<remotion:BocAllPropertiesPlaceholderColumnDefinition />

**with**

<remotion:BocSimpleColumnDefinition PropertyPathIdentifier="Street" />

<remotion:BocSimpleColumnDefinition PropertyPathIdentifier="Number" />

<remotion:BocSimpleColumnDefinition PropertyPathIdentifier="City" />

<remotion:BocSimpleColumnDefinition PropertyPathIdentifier="ZipCode" />

We have removed the Property DisplayName. We might also want to combine street with number.

1. **Replace in SearchResultLocationControl**

<remotion:BocSimpleColumnDefinition PropertyPathIdentifier="Street" />

<remotion:BocSimpleColumnDefinition PropertyPathIdentifier="Number" />

**with**

<remotion:BocCompoundColumnDefinition ColumnTitle="Name" ItemID="CompoundEdit"

FormatString="{0}, {1}">

<propertypathbindings>

<remotion:PropertyPathBinding PropertyPathIdentifier="Street" />

<remotion:PropertyPathBinding PropertyPathIdentifier="Number" />

</propertypathbindings>

<persistedcommand>

<remotion:BocListItemCommand Type="Event"/>

</persistedcommand>

</remotion:BocCompoundColumnDefinition>

Let’s do the same with EditPersonControl to get more practice.

1. **Replace in EditPersonControl.ascx**

<FixedColumns>

<remotion:BocAllPropertiesPlaceholderColumnDefinition />

<remotion:BocRowEditModeColumnDefinition CancelText="$res:Cancel" EditText="$res:Edit" SaveText="$res:Save" />

</FixedColumns>

<ListMenuItems>

<remotion:BocMenuItem ItemID="AddMenuItem" Text="$res:Add" />

</ListMenuItems>

1. **with**

<FixedColumns>

<remotion:BocRowEditModeColumnDefinition CancelText="$res:Cancel" EditText="$res:Edit"

SaveText="$res:Save" />

<remotion:BocSimpleColumnDefinition EnableIcon="False"

PropertyPathIdentifier="CountryCode">

<persistedcommand>

<remotion:BocListItemCommand />

</persistedcommand>

</remotion:BocSimpleColumnDefinition>

<remotion:BocSimpleColumnDefinition EnableIcon="False"

PropertyPathIdentifier="AreaCode">

<persistedcommand>

<remotion:BocListItemCommand />

</persistedcommand>

</remotion:BocSimpleColumnDefinition>

<remotion:BocSimpleColumnDefinition EnableIcon="False"

PropertyPathIdentifier="Number">

<persistedcommand>

<remotion:BocListItemCommand />

</persistedcommand>

</remotion:BocSimpleColumnDefinition>

<remotion:BocSimpleColumnDefinition EnableIcon="False"

PropertyPathIdentifier="Extension">

<persistedcommand>

<remotion:BocListItemCommand />

</persistedcommand>

</remotion:BocSimpleColumnDefinition>

</FixedColumns>

<ListMenuItems>

<remotion:BocMenuItem ItemID="AddMenuItem" Text="$res:Add" >

<PersistedCommand>

<remotion:BocMenuItemCommand></remotion:BocMenuItemCommand>

</PersistedCommand>

</remotion:BocMenuItem>

</ListMenuItems>

### Questions and Discussions

* If you compare this strategy to XAML, what are the advantages and disadvantages?
* In the Step 4 we added some additional markup stuff. What out what this additional changes might do!

### Task 3: Add Icons to Controls

We will add icons to our cells to show how to add services to our project.

1. **Add a File WebUiService.cs to the Web project in folder Classes**
2. **Paste the following code in WebUiService.cs**

using System;

using System.Web;

using Remotion.Data.DomainObjects;

using Remotion.ObjectBinding;

using Remotion.ObjectBinding.Web;

using Remotion.ObjectBinding.Web.UI.Controls;

using Remotion.Web.UI.Controls;

namespace PhoneBook.Web.Classes

{

public class WebUiService : IBusinessObjectWebUIService

{

public IconInfo GetIcon (IBusinessObject obj)

{

if (obj == null)

{

return new IconInfo ("~/Images/Icon-Null.gif", "16px", "16px");

}

else

{

// determine the static type of the

// passed object, i.e. the originally declared domain object

// class (Person, PhoneNumber or Location).

Type staticType = ((DomainObject) obj).GetPublicDomainObjectType ();

// assemble the path to the icon bitmap based on the class of the

// object

return new IconInfo ("~/Images/Icon-" + staticType.Name + ".gif", "16px", "16px");

}

}

public string GetToolTip (IBusinessObject obj)

{

return null;

}

public HelpInfo GetHelpInfo (IBusinessObjectBoundWebControl businessObjectBoundWebControl,

IBusinessObjectClass businessObjectClass,

IBusinessObjectProperty businessObjectProperty,

IBusinessObject businessObject)

{

return null;

}

}

}

1. **Add to global.asax.cs in Application\_Start**

BindableObjectProvider.GetProvider<BindableDomainObjectProviderAttribute> ().AddService (

typeof (IBusinessObjectWebUIService), new WebUiService ());

1. **Copy files such as icon-Location.gif into the Images folder.**

### Questions and Discussions

* Try to find out what the GetToolTip method does and what happens if you return a dummy string instead of null.

### Task 4: Add Delete Operations

We can already add information, but yet it is not possible to delete information again. This will be implemented in this task.

1. **Replace div in SearchResultLocationControl**

<div>

<remotion:BocList id="LocationList" runat="server" DataSourceControl="CurrentObject" OnListItemCommandClick="LocationList\_ListItemCommandClick">

<FixedColumns>

<remotion:BocSimpleColumnDefinition EnableIcon="True"

PropertyPathIdentifier="Street" ItemID="LeftColumnEdit">

<persistedcommand>

<remotion:BocListItemCommand Type="Event" />

</persistedcommand>

</remotion:BocSimpleColumnDefinition>

<remotion:BocSimpleColumnDefinition propertyPathIdentifier="Number" />

<remotion:BocSimpleColumnDefinition propertyPathIdentifier="City" />

<remotion:BocSimpleColumnDefinition propertyPathIdentifier="Country" />

<remotion:BocSimpleColumnDefinition propertyPathIdentifier="ZipCode" />

<remotion:BocCommandColumnDefinition ItemID="Edit" Text="$res:Edit">

<PersistedCommand>

<remotion:BocListItemCommand Type="Event" />

</PersistedCommand>

</remotion:BocCommandColumnDefinition>

<remotion:BocCommandColumnDefinition ItemID="Delete" Text="$res:Delete">

<PersistedCommand>

<remotion:BocListItemCommand Type="Event" />

</PersistedCommand>

</remotion:BocCommandColumnDefinition>

</FixedColumns>

</remotion:BocList>

</div>

1. **Adapt in the code file**

protected void LocationList\_ListItemCommandClick (object sender, BocListItemCommandClickEventArgs e)

{

if (e.Column.ItemID == "Edit" || e.Column.ItemID == "LeftColumnEdit")

{

try

{

EditLocationForm.Call (WxePage, (Location) e.BusinessObject);

ClientTransaction.Current.Commit ();

}

catch (WxeUserCancelException)

{

}

}

// BEGIN HANDLER

else if (e.Column.ItemID == "Delete")

{

((Location) e.BusinessObject).DeleteLocation ();

ClientTransaction.Current.Commit ();

var searchAllService = new BindableDomainObjectSearchAllService ();

var listLocations = searchAllService.GetAllObjects (ClientTransaction.Current, typeof (Location));

LocationList.LoadUnboundValue (listLocations, IsPostBack);

}

// END HANDLER

}

1. **Add to Location**

public Person[] FindPersons ()

{

var query = from p in QueryFactory.CreateLinqQuery<Person> ()

where p.Location == this

select p;

return query.ToArray ();

}

public void DeleteLocation ()

{

var persons = FindPersons ();

foreach (var p in persons)

{

p.Location = null;

}

Delete ();

ClientTransaction.Current.Commit ();

}

1. **Add to reference to remotion.linq in your domain project, if not already done so**

### Questions and Discussions

* The delete Method is protected in the base class. What do you think is the reason for this?

### Task 5: Add Customized Cells

In this sample we add a customized call, so that all phone numbers are displayed in a row.

1. **Adapt in SearchResultPersonControl**

<div>

<remotion:BocList id="PersonList" runat="server" DataSourceControl="CurrentObject" OnListItemCommandClick="PersonList\_ListItemCommandClick">

<FixedColumns>

<remotion:BocCompoundColumnDefinition ColumnTitle="Name" ItemID="CompoundEdit"

FormatString="{0}, {1}">

<propertypathbindings>

<remotion:PropertyPathBinding PropertyPathIdentifier="LastName" />

<remotion:PropertyPathBinding PropertyPathIdentifier="FirstName" />

</propertypathbindings>

<persistedcommand>

<remotion:BocListItemCommand Type="Event"/>

</persistedcommand>

</remotion:BocCompoundColumnDefinition>

<remotion:BocSimpleColumnDefinition EnableIcon="False"

PropertyPathIdentifier="Location">

<persistedcommand>

<remotion:BocListItemCommand />

</persistedcommand>

</remotion:BocSimpleColumnDefinition>

<remotion:BocCustomColumnDefinition

PropertyPathIdentifier="PhoneNumbers" CustomCellType="PhoneBook.Web.Classes.PhoneNumberCell"

CustomCellArgument="MaxPhoneNumbers=3,Commit=true">

</remotion:BocCustomColumnDefinition>

<remotion:BocCommandColumnDefinition ItemID="Edit" Text="$res:Edit">

<PersistedCommand>

<remotion:BocListItemCommand Type="Event" />

</PersistedCommand>

</remotion:BocCommandColumnDefinition>

</FixedColumns>

</remotion:BocList>

</div>

1. **Change in SearchResultPersonControl.cs**

protected void PersonList\_ListItemCommandClick (object sender, BocListItemCommandClickEventArgs e)

{

if (e.Column.ItemID == "Edit" || e.Column.ItemID == "CompoundEdit")

1. **Add file PhoneNumberCell.cs. to PhoneBook.Web.Classes**

using System;

using System.Collections;

using System.Web;

using System.Web.UI;

using PhoneBook.Domain;

using PhoneBook.Web.UI;

using Remotion.Data.DomainObjects;

using Remotion.Data.DomainObjects.ObjectBinding;

using Remotion.ObjectBinding.Web.UI.Controls;

using Remotion.Web.ExecutionEngine;

namespace PhoneBook.Web.Classes

{

public class PhoneNumberCell : BocCustomColumnDefinitionCell

{

public int MaxPhoneNumbers { get; set; }

public bool Commit { get; set; }

protected override void Render (HtmlTextWriter writer, BocCustomCellRenderArguments arguments)

{

var propertyPath = arguments.ColumnDefinition.GetPropertyPath();

var bo = arguments.BusinessObject;

var phoneNumbers = (IList) propertyPath.GetValue (bo, false, true);

for (int i = 0; i < MaxPhoneNumbers && i < phoneNumbers.Count; ++i)

{

var phoneNumber = (BindableDomainObject) phoneNumbers[i];

string renderedLink = String.Format ("<a href=\"#\" onclick=\"{0}\">{1}</a>",

GetPostBackClientEvent (phoneNumber.ID.ToString()),

HttpUtility.HtmlEncode (phoneNumber.DisplayName));

writer.Write (renderedLink);

writer.Write ("<br>");

}

}

protected override void OnClick (BocCustomCellClickArguments arguments, string eventArgument)

{

try

{

var id = ObjectID.Parse (eventArgument);

var page = (IWxePage) arguments.List.Page;

PhoneNumber number = PhoneNumber.GetObject (id);

var externalOption = new WxeCallOptionsExternal ("\_blank");

var externalOptionArgument = new WxeCallArguments ((Control) page, externalOption);

EditPhoneNumberForm.Call (page, externalOptionArgument, number);

if (Commit)

{

ClientTransaction.Current.Commit();

}

}

catch (WxeIgnorableException) { }

}

}

}

1. **Test and verify**

### Task 6: Adding LocationPicker to Person Page

In the next tasks we want to add the options to add locations in the Persons Page.

1. **Replace in EditPersonControl**

<remotion:BocReferenceValue ID="LocationField" runat="server" DataSourceControl="CurrentObject"

PropertyIdentifier="Location" >

</remotion:BocReferenceValue>

**With**

<remotion:BocReferenceValue ID="LocationField" runat="server"

DataSourceControl="CurrentObject" PropertyIdentifier="Location"

OnMenuItemClick = "LocationField\_MenuItemClick"

OptionsTitle="$res:OptionsActions">

<PersistedCommand>

<remotion:BocCommand></remotion:BocCommand>

</PersistedCommand>

<OptionsMenuItems>

<remotion:BocMenuItem Text="$res:NewLocation" ItemID="NewLocation">

</remotion:BocMenuItem>

<remotion:BocMenuItem Text="$res:PickLocation" ItemID="PickLocation">

</remotion:BocMenuItem>

</OptionsMenuItems>

1. **Add the corresponding EventHandler of the EditPersonControl.ascx.cs**

protected void LocationField\_MenuItemClick (object sender,

Remotion.Web.UI.Controls.WebMenuItemClickEventArgs e)

{

try

{

switch (e.Item.ItemID)

{

case "NewLocation":

LocationField.Value = Location.GetObject(EditLocationForm.Call (WxePage, null));

break;

case "PickLocation":

LocationField.Value = PickLocation.Call (WxePage);

break;

}

}

catch (WxeIgnorableException)

{

// fall back to here

}

}

1. **We have to add a new page to the GUI**

**PickLocation.aspx**

<%@ Page Language="C#" CodeBehind="PickLocation.aspx.cs" Inherits="PhoneBook.Web.UI.PickLocation" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html>

<head id="PickALocationHeader" runat="server">

<title><!-- Page title set in Page\_Load !--></title>

</head>

<form id="PickLocationForm" runat="server">

<remotion:FormGridManager ID="FormGridManager" runat="server" />

<remotion:BindableObjectDataSourceControl ID="PickLocationDataSource"

runat="server" Mode="Search" Type="PhoneBook.Domain.Location, PhoneBook.Domain" />

<table id="CountryFormGrid" runat="server">

<tr>

<td>

<remotion:SmartLabel runat="server" id="CountryLabel" ForControl="CountryField" />

</td>

<td>

<remotion:BocEnumValue ID="CountryField" runat="server"

DataSourceControl="PickLocationDataSource" PropertyIdentifier="Country" />

</td>

<td>

<remotion:WebButton ID="SearchButton" runat="server" onclick="SearchButton\_Click" Text="$res:Search" />

</td>

<td>

<remotion:WebButton ID="CancelButton" runat="server" onclick="CancelButton\_Click" Text="$res:Cancel" />

</td>

</tr>

</table>

<remotion:BocList ID="FilteredLocationsList" runat="server"

DataSourceControl="PickLocationDataSource" OnListItemCommandClick="FilteredLocationsList\_ListItemCommandClick">

<FixedColumns>

<remotion:BocSimpleColumnDefinition

PropertyPathIdentifier="Street" ItemID="LocationPick">

<persistedcommand>

<remotion:BocListItemCommand Type="Event" />

</persistedcommand>

</remotion:BocSimpleColumnDefinition>

<remotion:BocSimpleColumnDefinition PropertyPathIdentifier="Number" />

<remotion:BocSimpleColumnDefinition PropertyPathIdentifier="City" />

<remotion:BocSimpleColumnDefinition PropertyPathIdentifier="Country" />

<remotion:BocSimpleColumnDefinition PropertyPathIdentifier="ZipCode" />

</FixedColumns>

</remotion:BocList>

</form>

</html>

**PickLocation.aspx.cs**

using System;

using System.Linq;

using PhoneBook.Web.Classes;

using Remotion.Data.DomainObjects.Queries;

using Remotion.ObjectBinding.Web.UI.Controls;

using Remotion.Web.ExecutionEngine;

using Remotion.Web.UI.Globalization;

using PhoneBook.Domain;

namespace PhoneBook.Web.UI

{

// <WxeFunction>

// <ReturnValue type="Location" />

// <Variable name="items" type="Location[]" />

// </WxeFunction>

public partial class PickLocation : BasePage

{

protected void Page\_Load (object sender, EventArgs e)

{

Title = ResourceManagerUtility.GetResourceManager (this).GetString ("Pick~Location");

if (IsPostBack)

{

FilteredLocationsList.LoadUnboundValue (items, true);

}

}

protected void FilteredLocationsList\_ListItemCommandClick (object sender, BocListItemCommandClickEventArgs e)

{

if (e.Column.ItemID == "LocationPick")

{

ReturnValue = (Location) e.BusinessObject;

Return ();

}

else

{

throw new ArgumentException ();

}

}

protected void SearchButton\_Click (object sender, EventArgs e)

{

var locations = from loc in QueryFactory.CreateLinqQuery<Location> ()

where loc.Country == (Country?) CountryField.Value

select loc;

items = locations.ToArray ();

FilteredLocationsList.LoadUnboundValue (items, false);

}

protected void CancelButton\_Click (object sender, EventArgs e)

{

throw new WxeUserCancelException ();

}

}

1. **The next step is to run into a build problem to learn about our navigation system. Investigate the build error! And remember what we said about re-call!**
2. **Now add in EditLocationForm.aspx.cs a return type**

// <WxeFunction>

// <Parameter name="obj" type="Location" />

// <ReturnValue type="ObjectID" />

// </WxeFunction>

1. **Adapt SaveButton\_Click hander in EditLocationForm.aspx.cs**

protected void SaveButton\_Click (object sender, EventArgs e)

{

if (SaveObject ())

{

ReturnValue = obj.ID;

Return();

}

}

1. **Build two times**
2. **Add OptionsActions to Globalisation**

### Questions and Discussions

* Why was it necessary to build the project two times?

# Final Questions

* Discuss: Many people claim RIA technologies are better than web based technologies. What is your opinion about this? Is it a good idea to replace ASP.NET based pages with new Silverlight pages?

# Lab Summary

You have learned

|  |
| --- |
| * Lab1: Using UIGEN to generate a web application project from the domain * Lab2: First steps with re-call, the transactional navigation between forms * Lab3: First steps with re-bind, the layer for data binding |

Now you might want to

|  |
| --- |
| * Learn about re-strict, the security component (no HOL available until now) * Learn more about UI principles. It is highly recommended that even as developer you have a basic understanding of UI principles. There are some very good books on this topic such as * <http://www.amazon.com/Designing-Mind-Understanding-Interface-ebook/dp/B003H3IOXM/ref=sr_1_2?ie=UTF8&qid=1297333053&sr=8-2> |
|  |