

Level 100 – mixins

Hands on Labs

Copyright notice: ©2011 rubicon informationstechnologie gmbh all rights reserved

Table of Contents

[1 Lab 1: First steps with mixins 1](#_Toc288643343)

[1.1 Goal 1](#_Toc288643344)

[1.2 Theory – Why mixins 1](#_Toc288643345)

[1.3 Exercise 1 – ten minutes mixin 1](#_Toc288643346)

[1.4 Questions 3](#_Toc288643347)

[1.5 Free Exercise 3](#_Toc288643348)

[1.6 Exercise 2 – Parameter 3](#_Toc288643349)

[1.6.1 Questions 4](#_Toc288643350)

[1.7 Exercise 3 – A use case for „use“ 4](#_Toc288643351)

[1.7.1 Questions 4](#_Toc288643352)

[2 Lab Summary 5](#_Toc288643353)

[3 Common Practices 5](#_Toc288643354)

[3.1 Getting re-mix 5](#_Toc288643355)

# Lab 1: First steps with mixins

## Goal

Estimated Time: 30 Minutes

Goal of this HOL is to teach how to use mixins by programming examples and to highlight some advantages of using mixins. All exercises are created to be solved step by step. You should be able to finish each exercise with an executable result.

There are sections with question in this lab. Some answers to some questions might be straight forward. Other questions could be tricky. In case you are doing this exercise with other software developers, we encourage you to talk about your ideas and your proposed solutions.

After this exercise, you will be able to understand the advantages of mixins. You will be able to decide when a mixin is an appropriate solution to a problem and when not. You will be ready to bring a lot of new possibilities to your project by implementing mixins at the right place.

Please also have a look at the further readings section in the end of this HOL if you want to dig deeper.

Topics covered:

|  |
| --- |
| * Basics: What are mixins * Developing your first mixin * Understanding the difference between “use” and “extends” implementations * Using Parameters |

**Important:**

For this HOL you must have Visual Studio 2010 installed on your working PC. We recommend installing additional third party tools such as Jetbrains Resharper (<http://www.jetbrains.com/resharper/>) or Redgate Reflector (<http://reflector.red-gate.com/Download.aspx>) or equivalent tools on your working PC too. This will help you to understand some samples better and you are able to dig deeper.

For all examples, the re-mix binaries are required. You can either build them or download them via the build page (<http://remix.codeplex.com>)

## Theory – Why mixins

It is essential that you understand the purpose of mixins before you implement your first mixins. The discussion on why mixins are a powerful extension to .NET development is out of scope for this document.

If you are completely new to mixins, it is recommended that you read one or two basic articles on mixins before you get started.

## Exercise 1 – Ten minutes mixin

1. Start Visual Studio 2010
2. Add a new project and select a class library. For this HOL, we recommend the following settings:

Name: Domain

Location: C:\HOL\mixins

Solution name: HOLMixin

1. Create a new console application
2. Add the following references to the domain project

* remotion.dll
* remotion.interfaces.dll

You might want to read the section “Adding references” in case you are not sure where to put the binary assemblies.

1. Add a reference from the console app to the class library project
2. Add a file person.cs to the Domain project and use the following source code

using System;

namespace Domain

{

public class Person

{

public string FirstName { get; set; }

public string LastName { get; set; }

public DateTime BirthDay { get; set; }

}

}

1. Add a file EmployeeMixin.cs the domain and use the following source code

using System;

using Remotion.Mixins;

namespace Domain

{

public interface IEmployee

{

int Salary { get; set; }

DateTime? HireDate { get; set; }

}

[Extends(typeof(Person))]

public class EmployeeMixin : IEmployee

{

public int Salary { get; set; }

public DateTime? HireDate { get; set;}

public EmployeeMixin()

{

// set default values

Salary = 1000;

HireDate = DateTime.Now;

}

}

}

1. Add the following source code to the Console Application.

using Domain;

using Remotion.Reflection;

using Remotion.Mixins;

namespace ConsoleApp

{

class Program

{

static void Main(string[] args)

{

var employee = (IEmployee)ObjectFactory.Create<Person>(ParamList.Empty);

((Person) employee).FirstName = "Max";

((Person)employee).LastName = "Mustermann";

System.Console.WriteLine("Fullname: {0}", ((Person)employee).FirstName + " " + ((Person)employee).LastName);

System.Console.WriteLine("Salary: {0}", employee.Salary);

System.Console.ReadKey();

}

}

}

1. Debug and have a look at the result.

Within a very short time, you have learned that is very easy to implement a basic mixin within your solution. In the upcoming tasks, we want to dig deeper and explore some scenarios.

## Questions

* In the first sample, we implemented an Employee mixin for Person. Think about this implementation. If being an employee in a company is a context, what additional contexts could be there in the real world?
* You will probably find a lot of different contexts. How would you implement this with pure OOP? What are disadvantages and advantages over mixins?
* In the first sample, the Employee mixin extends the Person class. Does it make sense to apply this mixin to a different class? If not, try to give a reason.
* After thinking about the answer to the question before. It seems that Person is “a fixed constant” and this class will be extended by mixins. Can you think of a scenario where the mixin is a bundle of functionality that can be added to more than just one class?

## Free Exercise

* Add two mixins for different contexts to our sample.
* In the first context, the Person is a bank customer. Add at least an account number and a balance as properties for the mixins.
* In the second context, we want to manage a family. We should add composition with a mixin. As a family, we have a list of children containing 0 or more entries. Also add a spouse (which might of course be null too), a father, and a mother.
* How about required parameters? Can or should mother and father be a required parameter for the “FamilyMemberMixin”? Discuss!

## Exercise 2 – Parameter

In the free exercise, we have added more mixins.

At the current state, we would have to set the properties of the base class Person (such as FirstName) after its creation via the object factory manually.

The next step is to add parameters to the at the object factory.

1. Please add a new constructor with parameters to the person class. We also have to define a default constructor

public class Person

{

public Person(string firstName, string lastName)

{

FirstName = firstName;

LastName = lastName;

}

public Person()

{

}

public string FirstName { get; set; }

public string LastName { get; set; }

public DateTime BirthDay { get; set; }

1. Replace

var employee = (IFamilyMember)ObjectFactory.Create<Person>(ParamList.Empty);

with

var employee = (IEmployee)ObjectFactory.Create<Person>( "Max", "Mustermann");

### Questions

* In the first example, we have not specified a default constructor. When we now add a copy constructor without adding a default constructor too, we run into a compiler error. Why?
* In our sample, objects are created via a Factory Design Pattern. Can you explain the advantages of using Factories to instantiate classes over constructors within a few sentences? How would you try to convince a fellow developer to use factories instead of construction? Does this approach does also have disadvantages?
* If you use intellisense to look for possible parameters for ParamList.Create(), you will find that there are a lot of possibilities to choose. Can you explain the reason?

## Exercise 3 – A use case for „use“

Our first sample is a “extends” scenario. We have a fixed base class that can be extended with one or more mixins. There is also another scenario, where we have “a fixed mixin” that can be added to various classes. This scenario is called “use” scenario. In the following example we implement a mixin that will generate a string that summarizes all public properties of the class and their current values.

This mixin could be added to our sample. As we might want to add information to the console such as:

System.Console.WriteLine("Fullname: {0}", ((Person)employee).FirstName + " " + ((Person)employee).LastName);

The idea is to generate a string with all public properties in a function (we will call it PropertyValues of a mixin. This mixin is then used by various classes.

System.Console.WriteLine(((IPropertyValue)employee).PropertyValues);

1. We add a new Mixin. In the Property PropertyValues we read all properties and

public interface IPropertyValueProvider

{

string PropertyValues();

}

public class PropertyValueStringChainMixin<T> : Mixin<T>, IPropertyValueProvider where T : class

{

public string PropertyValues()

{

StringBuilder stringBuilder = new StringBuilder();

foreach (PropertyInfo propertyInfo in This.GetType().GetProperties())

{

stringBuilder.Append(propertyInfo.Name);

stringBuilder.Append(": ");

stringBuilder.Append(propertyInfo.GetValue(This, null));

stringBuilder.AppendLine();

}

return stringBuilder.ToString();

}

}

}

1. We also add the attribute to the class declaration of Person.

[Uses(typeof(PropertyValueStringChainMixin<Person>))]

### Questions

* We would not need to add a mixin to achieve this result. We could also add this method to the class Person too. What are the advantages to using mixins?
* Could you get the same advantages also with extension methods in this scenario?
* You might want to try out different parameters in the GetProperties method, such as

GetProperties(

BindingFlags.Public|BindingFlags.NonPublic|BindingFlags.Instance))

What happens?

* If you have tried it out, you will run into a StackOverflowExpection. Adding if (propertyInfo.Name != "PropertyValues") after the for clause does not solve the problem. Why?
* If you have solved the problem via if (!propertyInfo.Name.Contains("PropertyValues")) or something similar, you will see that the Salary is always 1000. Why?
* Can you explain what the following constructor does in detail? What is <T>?

public class PropertyValueStringChainMixin<T> : Mixin<T>, IPropertyValue where T : class

* Do you know what the keyword where means?
* Is it possible to change the constructor in the following way?

public class PropertyValueStringChainMixin : Mixin<object>, IPropertyValueProvider

# Lab Summary

In this lab, you have successfully done the following

|  |
| --- |
| * You have created a mixin sample, where you have extended a class Person * You have used a mixin within the Person sample to collect property information * You have added parameter list to the construction. |

As you now are experienced in the main usage of mixins, you might want to dig deeper to become a real mixin expert. We suggest:

* Read “Head First Design Patterns” to be able to understand design techniques in total better. It helps you to put mixins in relation to other design topics
* Read the next re-motion HOL to understand another re-motion technology.
* On <http://en.wikipedia.org/wiki/Mixin> in further readings, there are many references to other mixin implementations, you might want to read them and compare them with the re-motion approach.
* Read the blogs on [www.re-motion.org](http://www.re-motion.org)
* For those who want to become real experts: Read “CLR via C#”, if you want to understand more in how code is generated and how mixins are generated. Take care: Reading the most essential parts book also with good background experience may take a lot of time. Plan this for an extended research time and prepare enough coffee.