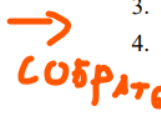



Документ Практическое_руководство_WPF_24

Практическая работа 7, упражнение 2 (Создание и использование словаря ресурсов)

Перед пунктом 5, где производится использование созданного (1-4) словаря ресурсов, не хватает пункта про построение приложения, поскольку файл xaml не считывает созданный ресурс и соответственно указывает на ошибку.

-  
- ```
</Style>
```
3. Сохраните измененный файл.
4. В файле App.xaml, добавьте объект `<ResourceDictionary.MergedDictionaries>` и в нем укажите ссылки на желаемую тему и ваш словарь ресурсов (именно в такой последовательности, чтобы ваши стили перекрыли стили стандартной темы):
- ```
<Application.Resources>
  <ResourceDictionary>
    <ResourceDictionary.MergedDictionaries>
      <ResourceDictionary Source="ShinyBlue.xaml"/>
      <ResourceDictionary Source="MyDictionary.xaml"/>
    </ResourceDictionary.MergedDictionaries>
  </ResourceDictionary>
</Application.Resources>
```
5. Для применения указанных стилей и шаблонов в словаре ресурсов:
- В файлах XAML главного окна и второго окна добавьте после установки размеров окон свойство цвета фона и присвойте ему ресурс градиентной заливки:
`Background="{StaticResource gradientStyle}"`
 - В файле XAML второго окна в свойствах кнопки укажите атрибут – ссылку на созданный шаблон `Template="{DynamicResource customButtonTemplate}"`.
6. Постройте и протестируйте приложение. Должны работать настройки вашего словаря ресурсов (поведение и вид текстовых полей обоих окон, а также кнопка второго окна) и стандартной темы (остальные кнопки и другие элементы).

Практическая работа 8, упражнение 1 (Создание привязки данных к объекту-коллекции)

Также после пунктов про создание новых классов не хватило пункта про сборку, чтобы в xaml корректно подключались созданное пространство и классы.

ооновлении всего списка.

5. В этот же файл добавьте новый класс **StudentList** производный от класса **ObservableCollection<Student>**:

```
public class StudentList : ObservableCollection<Student>
{
    public StudentList()
    {
        Add(new Student("Lorin Kanev", true));
        Add(new Student("Ivan Petrov", true));
        Add(new Student("Sergey Masov", false));
        Add(new Student("Tais Frolova", true));
        Add(new Student("Elena Diva", false));
    }
}
```

→
или

6. Подключите требуемое пространство имен **System.Collections.ObjectModel**.

7. Добавьте во второе окно (MyWindow) в перечень пространств имен ссылку на пространство имен CLR (у вас может быть другое имя пространства имен, в котором объявлены новые классы): `xmlns:Local="clr-namespace:WpfApplication4"`.

→ СУБРАТЪ

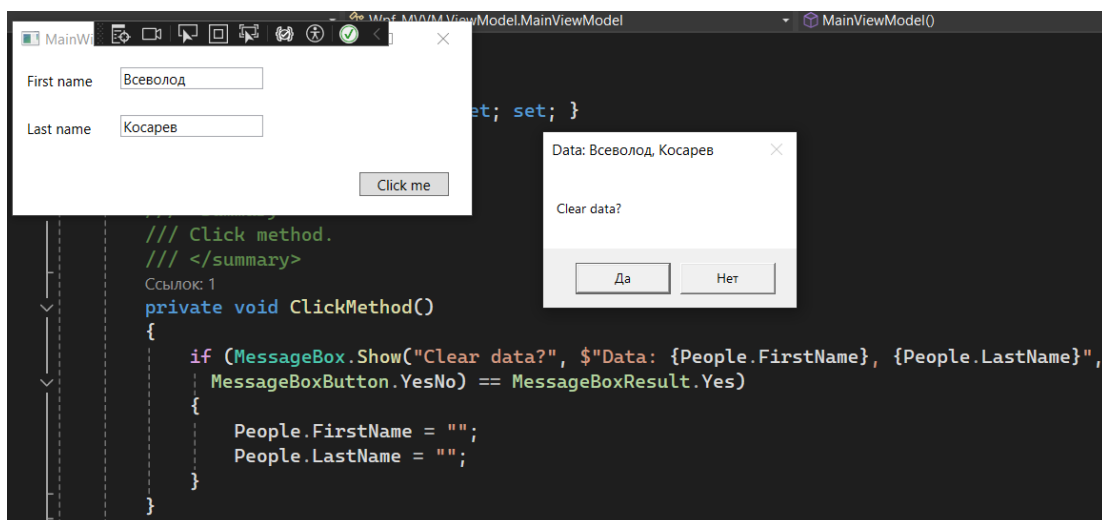
Указание привязки **ListBox** к источнику данных

1. Сделайте коллекцию доступной для привязки тем же способом, который используется для привязки других объектов CLR, для чего создайте экземпляр коллекции в XAML и укажите коллекцию в качестве ресурса (раздел ресурса текущего окна, можно перед закрывающимся тегом `</Window.Resources>`):
`<Local:StudentList x:Key="studentCollection"/>`
2. В списке `<ListBox x:Name="listBox1"` выполните привязку к коллекции, добавив

Лабораторная работа 2, упражнение 4 (Настройка обновления свойств)

Не было иллюстрации того, что при изменении свойств объекта автоматически будет обновляться форма. То есть сделали привязку, но она ни на что не повлияла.

В своей реализации я сделал вызов окна, которое спрашивает, нужно ли произвести очистку данных. Если согласиться с очисткой, тогда будут выполнены изменения в свойствах и это соответственно повлияет на форму, и она обновится с пустыми значениями.



Руководство_ASP.NET_MVC_O_22

Лабораторная работа 1, упражнение 1 (Создание веб-приложения ASP.NET MVC на основе контроллера)

Здесь не было информации по тому, как настраивать проект под Program.cs, но это не сильно мешало и, возможно, в версии 23 года уже дописано

Лабораторная работа 6, упражнение 1 (Добавление шаблонного контроллера)

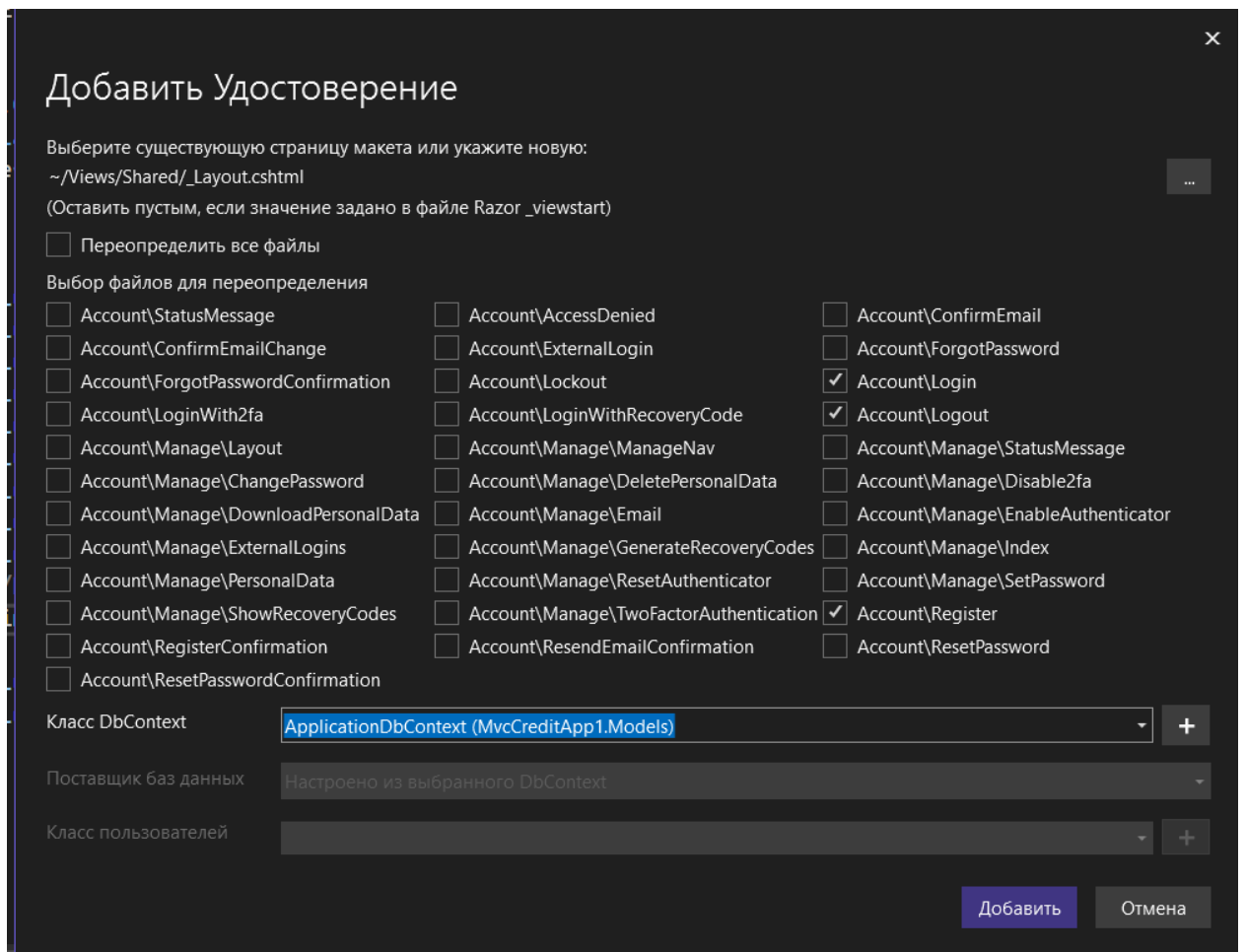
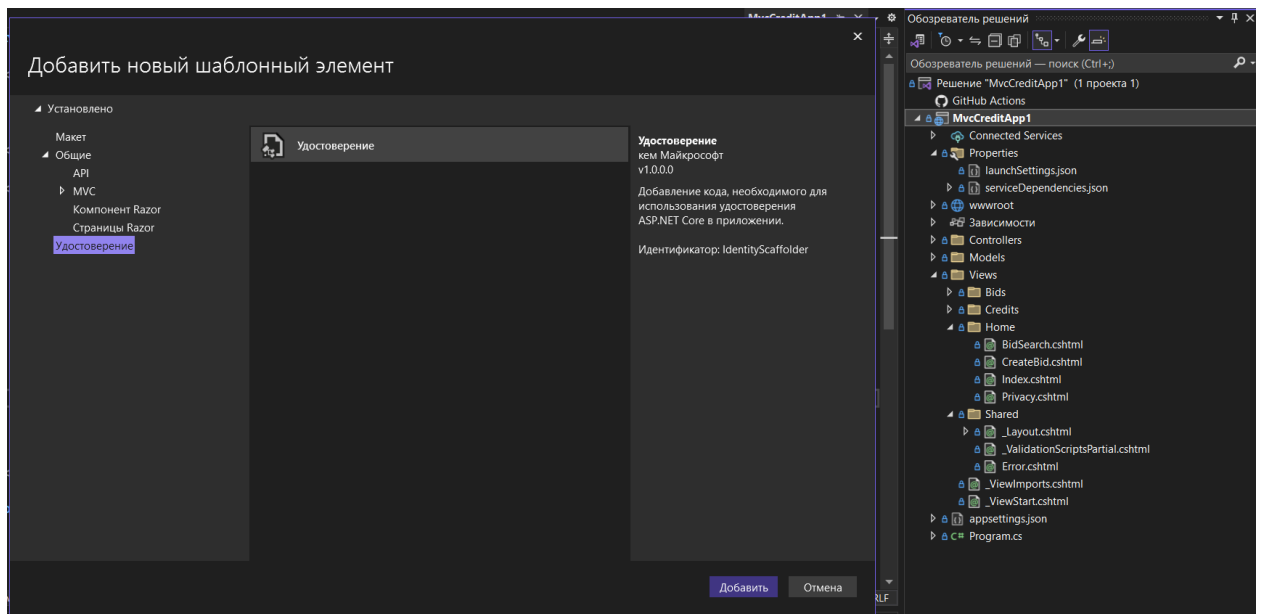
Из-за того, что в net8.0 начали использовать новый подход к контексту данных нужно было либо пробовать создавать контроллер с новым классом контекста и адаптировать информацию из документа к нему, либо создавать контроллер, а затем удалять новый класс и переписывать весь код контроллера к созданному ранее классу контекста. Я пробовал оба способа и сработал только второй из-за того, что я не знал нюанса о работе только в асинхронном режиме в новых контекстах.

Лабораторная работа 7, упражнение 1 (Использование AJAX и частичных страниц)

При подключении пакета не было папки Scripts, чтобы проверить наличие скриптов. Не получилось зарегистрировать Unobtrusive AJAX без использования ещё одного пакета LigerShark.WebOptimizer.Core, но всё равно не удалось внедрить скрипты AJAX в View, потому что Ajax не распознавался в контексте скрипта Index.cshtml.

Лабораторная работа 8, упражнение 1 (Использование типа аутентификации Individual User Accounts)

Не удалось настроить авторизацию, так как она не была заранее внедрена в актуальный шаблон веб-приложения ASP.NET, поэтому пробовал добавить удостоверение через IdentityScaffolded, но постоянно возникала ошибка на этапе генерации кода:





Ошибка

При запуске выбранного генератора кода произошла ошибка:

"Failed to compile the project in memory

C:\Users\iamvk\Desktop\ITMO\Master degree\2 sem\Modern

programming technologies in infocommunication

systems\Tasks\MPT_in_IS_2024\ASP.NET

MVC\MvcCreditApp1\MvcCreditApp1\Program.cs(17,18): error

CS0121: Неоднозначный вызов следующих методов или свойств:

"Microsoft.Extensions.DependencyInjection.IdentityServiceCollectionEx
tensions.AddIdentity<TUser,

TRole>(Microsoft.Extensions.DependencyInjection.IServiceCollection,
System.Action<Microsoft.AspNetCore.Identity.IdentityOptions>)" и

"Microsoft.Extensions.DependencyInjection.IdentityServiceCollectionEx
tensions.AddIdentity<TUser,

TRole>(Microsoft.Extensions.DependencyInjection.IServiceCollection,
System.Action<Microsoft.AspNetCore.Identity.IdentityOptions>)"

OK