Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Ярославский государственный технический университет»

Кафедра «Информационные системы и технологии»

Работа защищена

с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_

Преподаватель

Прозоров А.Н.

«\_\_\_»\_\_\_\_\_\_\_\_\_\_ 2025

**РАЗРАБОТКА системЫ контроля времени пребывания детей на детской площадке НА ПЛАТФОРМЕ WPF**

Отчет о лабораторной работе по курсу

«Технология программирования»

ЯГТУ 09.03.02 - 004 ЛР

|  |  |
| --- | --- |
|  | Отчет выполнил  студент группы ЗЦИС-26  Прусакова Ю.Д.  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_ 2025 |

2025

**Цель работы**

Разработать приложение для системы контроля времени пребывания детей на детской площадке на платформе WPF.

Обеспечить работоспособность с базой данных на основе SQLite.

**Задание**

Разработать программу, предоставляющую пользователю систему контроля времени пребывания детей на детской площадке. Система должна обеспечивать следующие требования:

* формирование заявки "посещение", с указанием данных "заказчика" (родитель, ребенок), время начала посещения, время пребывания
* контроль выполнения сформированных заявок в реальном времени
* ведение истории посещения с возможностью формирования отчёта посещений за указанный период времени
* Обеспечение работы базы данных на основе SQLite и её взаимодействия с программой

**Ход выполнения задания**

При запуске приложения открывается рабочее окно.

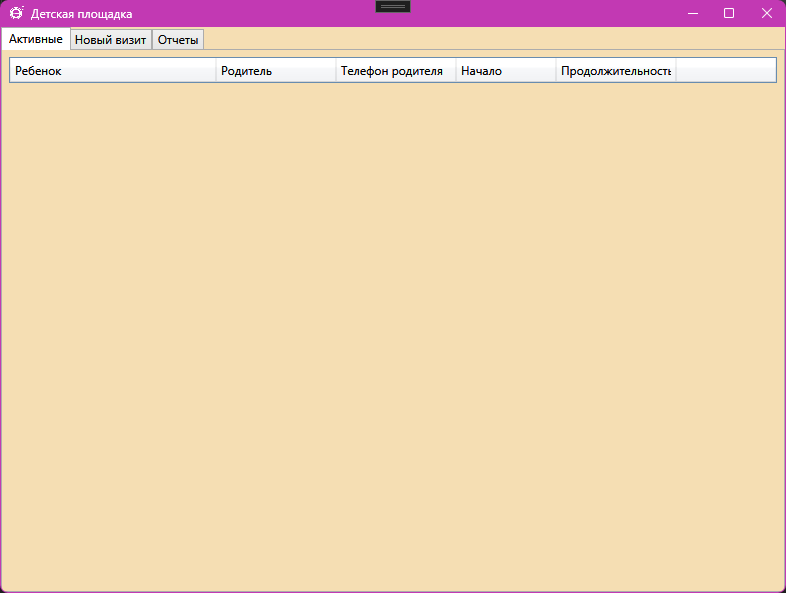


Рисунок 1 – Запуск приложения

Добавляем ребенка.

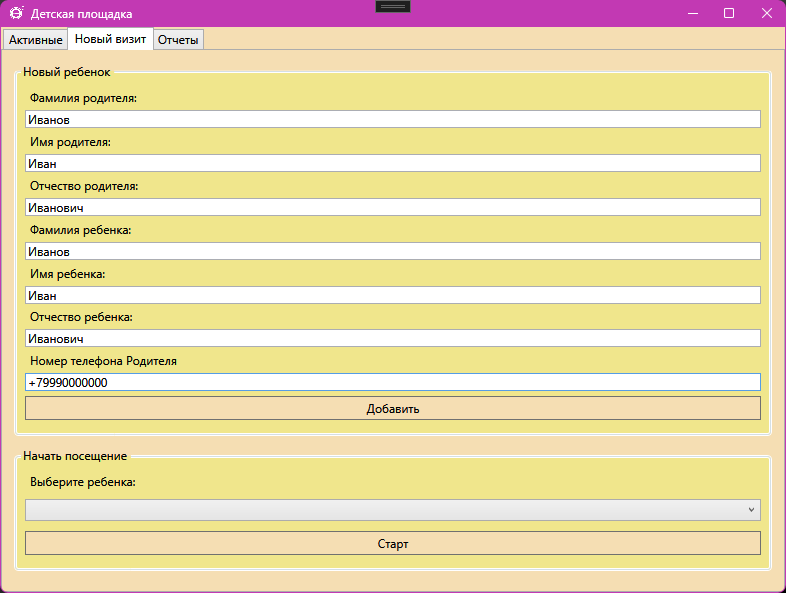


Рисунок 2 – Добавление ребенка

При добавлении ребенка, приложение предоставляет выбор ребенка из выпадающего списка.

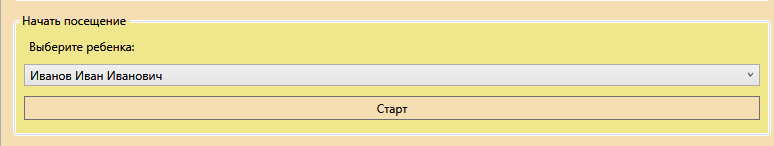


Рисунок 3 – Выбор ребенка из списка

Запуск таймера.

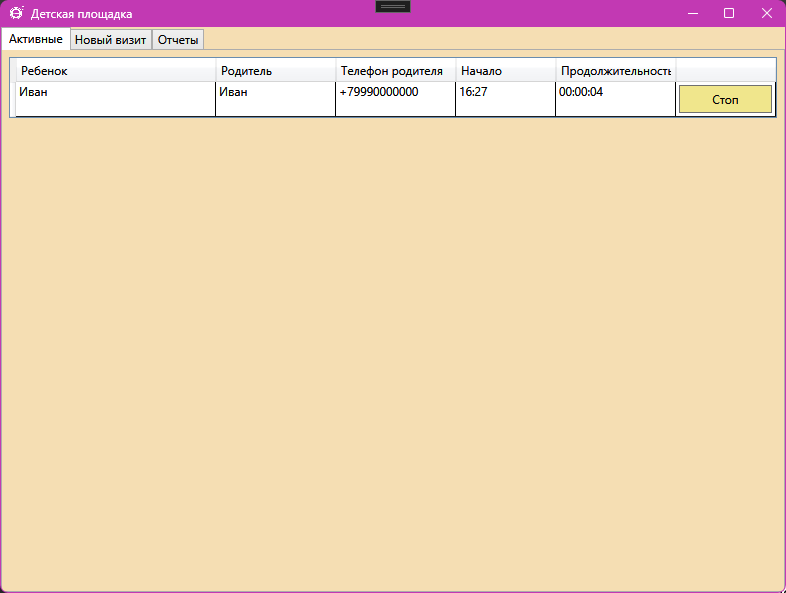


Рисунок 4 – Запуск таймера

Формирование отчета посещений.

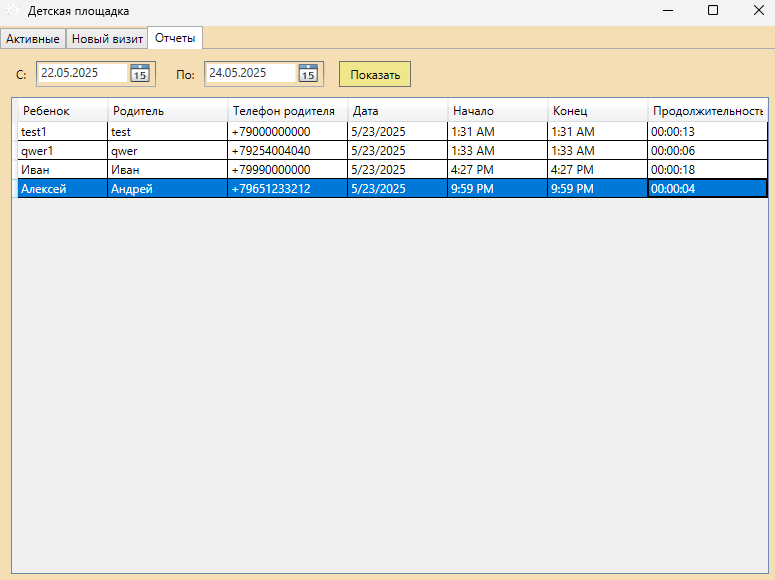


Рисунок 5- Отчет посещений

Схема базы данных.

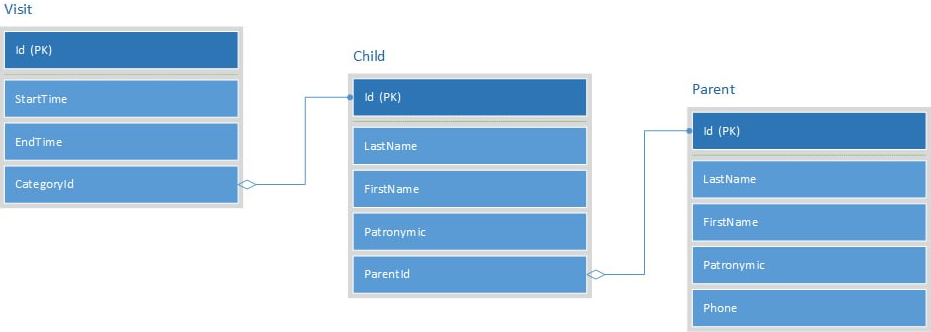


Рисунок 6 – База данных

**Вывод**

Была разработана программа, предоставляющая пользователю систему контроля времени пребывания детей на детской площадке. Система соответствует следующим требованиям:

* формирование заявки "посещение", с указанием данных "заказчика" (родитель, ребенок), время начала посещения, время пребывания
* контроль выполнения сформированных заявок в реальном времени
* ведение истории посещения с возможностью формирования отчёта посещений за указанный период времени
* Обеспечение работы базы данных на основе SQLite и её взаимодействия с программой

**Приложение А**

**Листинг DataBase.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using System.Data.Entity;

using System.Windows.Threading;

using System.Text.RegularExpressions;

namespace ChildPlaygroundTracker

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

private Database db = new Database();

private DispatcherTimer timer = new DispatcherTimer();

public MainWindow()

{

InitializeComponent();

LoadData();

timer.Interval = TimeSpan.FromSeconds(1);

timer.Tick += UpdateDurations;

timer.Start();

}

private void LoadData()

{

db.Children

.Include(c => c.Parent)

.Load();

db.Visits

.Include(v => v.Child.Parent)

.Load();

ChildrenCombo.ItemsSource = db.Children.Local;

ActiveVisitsGrid.ItemsSource = db.Visits.Local.Where(v => v.EndTime == null);

ReportGrid.ItemsSource = db.Visits.Local.Where(v => v.EndTime != null);

}

private void UpdateDurations(object sender, EventArgs e)

{

ActiveVisitsGrid.Items.Refresh();

}

private void AddChild\_Click(object sender, RoutedEventArgs e)

{

// Проверка заполненности полей родителя

if (string.IsNullOrWhiteSpace(ParentLastName.Text))

{

MessageBox.Show("Введите фамилию родителя!");

return;

}

if (string.IsNullOrWhiteSpace(ParentFirstName.Text))

{

MessageBox.Show("Введите имя родителя!");

return;

}

if (string.IsNullOrWhiteSpace(ParentPatronymic.Text))

{

MessageBox.Show("Введите отчество родителя!");

return;

}

// Проверка заполненности полей ребенка

if (string.IsNullOrWhiteSpace(ChildLastName.Text))

{

MessageBox.Show("Введите фамилию ребенка!");

return;

}

if (string.IsNullOrWhiteSpace(ChildFirstName.Text))

{

MessageBox.Show("Введите имя ребенка!");

return;

}

if (string.IsNullOrWhiteSpace(ChildPatronymic.Text))

{

MessageBox.Show("Введите отчество ребенка!");

return;

}

// Проверка телефона

var phoneRegex = new Regex(@"^\+7\d{10}$");

if (!phoneRegex.IsMatch(ParentPhone.Text.Trim()))

{

MessageBox.Show("Некорректный формат телефона!\nПример: +71234567890");

return;

}

// Проверка на существующий номер

if (db.Parents.Any(p => p.Phone == ParentPhone.Text.Trim()))

{

MessageBox.Show("Этот номер уже зарегистрирован!");

return;

}

// Создание родителя и ребенка

var parent = new Parent

{

LastName = ParentLastName.Text.Trim(),

FirstName = ParentFirstName.Text.Trim(),

Patronymic = ParentPatronymic.Text.Trim(),

Phone = ParentPhone.Text.Trim()

};

db.Parents.Add(parent);

db.SaveChanges(); // Сохраняем родителя, чтобы получить его Id

var child = new Child

{

LastName = ChildLastName.Text.Trim(),

FirstName = ChildFirstName.Text.Trim(),

Patronymic = ChildPatronymic.Text.Trim(),

ParentId = parent.Id

};

db.Children.Add(child);

db.SaveChanges();

// Очистка полей

ParentLastName.Clear();

ParentFirstName.Clear();

ParentPatronymic.Clear();

ChildLastName.Clear();

ChildFirstName.Clear();

ChildPatronymic.Clear();

ParentPhone.Clear();

// Обновление данных и сообщение

LoadData();

MessageBox.Show("Ребенок и родитель успешно добавлены!", "Успех", MessageBoxButton.OK, MessageBoxImage.Information);

}

private void StartVisit\_Click(object sender, RoutedEventArgs e)

{

if (ChildrenCombo.SelectedItem == null)

{

MessageBox.Show("Выберите ребенка!");

return;

}

var childId = ((Child)ChildrenCombo.SelectedItem).Id;

if (db.Visits.Any(v => v.ChildId == childId && v.EndTime == null))

{

MessageBox.Show("Этот ребенок уже на площадке!");

return;

}

var visit = new Visit

{

ChildId = childId,

StartTime = DateTime.Now

};

db.Visits.Add(visit);

db.SaveChanges();

// Сброс выбора и сообщение

ChildrenCombo.SelectedItem = null;

LoadData();

MessageBox.Show("Посещение начато!", "Успех", MessageBoxButton.OK, MessageBoxImage.Information);

}

private void StopVisit\_Click(object sender, RoutedEventArgs e)

{

var visit = (Visit)((FrameworkElement)sender).DataContext;

visit.EndTime = DateTime.Now;

db.SaveChanges();

LoadData();

}

private void GenerateReport\_Click(object sender, RoutedEventArgs e)

{

var start = StartDate.SelectedDate ?? DateTime.MinValue;

var end = EndDate.SelectedDate ?? DateTime.MaxValue;

ReportGrid.ItemsSource = db.Visits.Local

.Where(v => v.StartTime >= start && v.EndTime <= end)

.ToList();

}

}

}

**Приложение Б**

**Листинг MainWindow.xaml.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using System.Data.Entity;

using System.Windows.Threading;

using System.Text.RegularExpressions;

namespace ChildPlaygroundTracker

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

private Database db = new Database();

private DispatcherTimer timer = new DispatcherTimer();

public MainWindow()

{

InitializeComponent();

LoadData();

timer.Interval = TimeSpan.FromSeconds(1);

timer.Tick += UpdateDurations;

timer.Start();

}

private void LoadData()

{

db.Children

.Include(c => c.Parent)

.Load();

db.Visits

.Include(v => v.Child.Parent)

.Load();

ChildrenCombo.ItemsSource = db.Children.Local;

ActiveVisitsGrid.ItemsSource = db.Visits.Local.Where(v => v.EndTime == null)

ReportGrid.ItemsSource = db.Visits.Local.Where(v => v.EndTime != null);

}

private void UpdateDurations(object sender, EventArgs e)

{

ActiveVisitsGrid.Items.Refresh();

}

private void AddChild\_Click(object sender, RoutedEventArgs e)

{

// Проверка заполненности полей родителя

if (string.IsNullOrWhiteSpace(ParentLastName.Text))

{

MessageBox.Show("Введите фамилию родителя!");

return;

}

if (string.IsNullOrWhiteSpace(ParentFirstName.Text))

{

MessageBox.Show("Введите имя родителя!");

return;

}

if (string.IsNullOrWhiteSpace(ParentPatronymic.Text))

{

MessageBox.Show("Введите отчество родителя!");

return;

}

// Проверка заполненности полей ребенка

if (string.IsNullOrWhiteSpace(ChildLastName.Text))

{

MessageBox.Show("Введите фамилию ребенка!");

return;

}

if (string.IsNullOrWhiteSpace(ChildFirstName.Text))

{

MessageBox.Show("Введите имя ребенка!");

return;

}

if (string.IsNullOrWhiteSpace(ChildPatronymic.Text))

{

MessageBox.Show("Введите отчество ребенка!");

return;

}

// Проверка телефона

var phoneRegex = new Regex(@"^**\+**7\d{10}$");

if (!phoneRegex.IsMatch(ParentPhone.Text.Trim()))

{

MessageBox.Show("Некорректный формат телефона!\nПример: +71234567890");

return;

}

// Проверка на существующий номер

if (db.Parents.Any(p => p.Phone == ParentPhone.Text.Trim()))

{

MessageBox.Show("Этот номер уже зарегистрирован!");

return;

}

// Создание родителя и ребенка

var parent = new Parent

{

LastName = ParentLastName.Text.Trim(),

FirstName = ParentFirstName.Text.Trim(),

Patronymic = ParentPatronymic.Text.Trim(),

Phone = ParentPhone.Text.Trim()

};

db.Parents.Add(parent);

db.SaveChanges(); // Сохраняем родителя, чтобы получить его Id

var child = new Child

{

LastName = ChildLastName.Text.Trim(),

FirstName = ChildFirstName.Text.Trim(),

Patronymic = ChildPatronymic.Text.Trim(),

ParentId = parent.Id

};

db.Children.Add(child);

db.SaveChanges();

// Очистка полей

ParentLastName.Clear();

ParentFirstName.Clear();

ParentPatronymic.Clear();

ChildLastName.Clear();

ChildFirstName.Clear();

ChildPatronymic.Clear();

ParentPhone.Clear();

// Обновление данных и сообщение

LoadData();

MessageBox.Show("Ребенок и родитель успешно добавлены!", "Успех", MessageBoxButton.OK, MessageBoxImage.Information);

}

private void StartVisit\_Click(object sender, RoutedEventArgs e)

{

if (ChildrenCombo.SelectedItem == null)

{

MessageBox.Show("Выберите ребенка!");

return;

}

var childId = ((Child)ChildrenCombo.SelectedItem).Id;

if (db.Visits.Any(v => v.ChildId == childId && v.EndTime == null))

{

MessageBox.Show("Этот ребенок уже на площадке!");

return;

}

var visit = new Visit

{

ChildId = childId,

StartTime = DateTime.Now

};

db.Visits.Add(visit);

db.SaveChanges();

// Сброс выбора и сообщение

ChildrenCombo.SelectedItem = null;

LoadData();

MessageBox.Show("Посещение начато!", "Успех", MessageBoxButton.OK, MessageBoxImage.Information);

}

private void StopVisit\_Click(object sender, RoutedEventArgs e)

{

var visit = (Visit)((FrameworkElement)sender).DataContext;

visit.EndTime = DateTime.Now;

db.SaveChanges();

LoadData();

}

private void GenerateReport\_Click(object sender, RoutedEventArgs e)

{

var start = StartDate.SelectedDate ?? DateTime.MinValue;

var end = EndDate.SelectedDate ?? DateTime.MaxValue;

ReportGrid.ItemsSource = db.Visits.Local

.Where(v => v.StartTime >= start && v.EndTime <= end)

.ToList();

}

}

}

**Приложение В**

**Листинг WpfXamlLoader.cs**

using System.Collections;

using System.Windows.Baml2006;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Diagnostics;

using System.Windows.Navigation;

using System.Xaml;

using System.Xaml.Permissions;

using MS.Internal;

using MS.Internal.Xaml.Context;

using MS.Utility;

namespace System.Windows.Markup;

internal class WpfXamlLoader

{

private static Lazy<XamlMember> XmlSpace = new Lazy<XamlMember>(() => new WpfXamlMember(XmlAttributeProperties.XmlSpaceProperty, isAttachable: true));

public static object Load(System.Xaml.XamlReader xamlReader, bool skipJournaledProperties, Uri baseUri)

{

XamlObjectWriterSettings settings = XamlReader.CreateObjectWriterSettings();

object obj = Load(xamlReader, null, skipJournaledProperties, null, settings, baseUri);

EnsureXmlNamespaceMaps(obj, xamlReader.SchemaContext);

return obj;

}

public static object LoadDeferredContent(System.Xaml.XamlReader xamlReader, IXamlObjectWriterFactory writerFactory, bool skipJournaledProperties, object rootObject, XamlObjectWriterSettings parentSettings, Uri baseUri)

{

XamlObjectWriterSettings settings = XamlReader.CreateObjectWriterSettings(parentSettings);

return Load(xamlReader, writerFactory, skipJournaledProperties, rootObject, settings, baseUri);

}

public static object LoadBaml(System.Xaml.XamlReader xamlReader, bool skipJournaledProperties, object rootObject, XamlAccessLevel accessLevel, Uri baseUri)

{

XamlObjectWriterSettings xamlObjectWriterSettings = XamlReader.CreateObjectWriterSettingsForBaml();

xamlObjectWriterSettings.RootObjectInstance = rootObject;

xamlObjectWriterSettings.AccessLevel = accessLevel;

object obj = Load(xamlReader, null, skipJournaledProperties, rootObject, xamlObjectWriterSettings, baseUri);

EnsureXmlNamespaceMaps(obj, xamlReader.SchemaContext);

return obj;

}

internal static void EnsureXmlNamespaceMaps(object rootObject, XamlSchemaContext schemaContext)

{

if (rootObject is DependencyObject dependencyObject)

{

dependencyObject.SetValue(value: (schemaContext is XamlTypeMapper.XamlTypeMapperSchemaContext xamlTypeMapperSchemaContext) ? xamlTypeMapperSchemaContext.GetNamespaceMapHashList() : new Hashtable(), dp: XmlAttributeProperties.XmlNamespaceMapsProperty);

}

}

private static object Load(System.Xaml.XamlReader xamlReader, IXamlObjectWriterFactory writerFactory, bool skipJournaledProperties, object rootObject, XamlObjectWriterSettings settings, Uri baseUri)

{

XamlObjectWriter xamlObjectWriter = null;

MS.Internal.Xaml.Context.XamlContextStack<WpfXamlFrame> stack = new MS.Internal.Xaml.Context.XamlContextStack<WpfXamlFrame>(() => new WpfXamlFrame());

int persistId = 1;

settings.AfterBeginInitHandler = delegate(object sender, XamlObjectEventArgs args)

{

if (EventTrace.IsEnabled(EventTrace.Keyword.KeywordPerf | EventTrace.Keyword.KeywordXamlBaml, EventTrace.Level.Verbose))

{

IXamlLineInfo xamlLineInfo2 = xamlReader as IXamlLineInfo;

int num = -1;

int num2 = -1;

if (xamlLineInfo2 != null && xamlLineInfo2.HasLineInfo)

{

num = xamlLineInfo2.LineNumber;

num2 = xamlLineInfo2.LinePosition;

}

EventTrace.EventProvider.TraceEvent(EventTrace.Event.WClientParseXamlBamlInfo, EventTrace.Keyword.KeywordPerf | EventTrace.Keyword.KeywordXamlBaml, EventTrace.Level.Verbose, (args.Instance == null) ? 0 : PerfService.GetPerfElementID(args.Instance), num, num2);

}

if (args.Instance is UIElement uIElement)

{

uIElement.SetPersistId(persistId++);

}

XamlSourceInfoHelper.SetXamlSourceInfo(args.Instance, args, baseUri);

if (args.Instance is DependencyObject dependencyObject && stack.CurrentFrame.XmlnsDictionary != null)

{

XmlnsDictionary xmlnsDictionary = stack.CurrentFrame.XmlnsDictionary;

xmlnsDictionary.Seal();

XmlAttributeProperties.SetXmlnsDictionary(dependencyObject, xmlnsDictionary);

}

stack.CurrentFrame.Instance = args.Instance;

if (xamlReader is RestrictiveXamlXmlReader && args != null)

{

if (args.Instance is ResourceDictionary resourceDictionary)

{

resourceDictionary.IsUnsafe = true;

}

else if (args.Instance is Frame frame)

{

frame.NavigationService.IsUnsafe = true;

}

else if (args.Instance is NavigationWindow navigationWindow)

{

navigationWindow.NavigationService.IsUnsafe = true;

}

}

};

xamlObjectWriter = ((writerFactory == null) ? new XamlObjectWriter(xamlReader.SchemaContext, settings) : writerFactory.GetXamlObjectWriter(settings));

IXamlLineInfo xamlLineInfo = null;

try

{

xamlLineInfo = xamlReader as IXamlLineInfo;

IXamlLineInfoConsumer xamlLineInfoConsumer = xamlObjectWriter;

bool shouldPassLineNumberInfo = false;

if (xamlLineInfo != null && xamlLineInfo.HasLineInfo && xamlLineInfoConsumer != null && xamlLineInfoConsumer.ShouldProvideLineInfo)

{

shouldPassLineNumberInfo = true;

}

IStyleConnector styleConnector = rootObject as IStyleConnector;

TransformNodes(xamlReader, xamlObjectWriter, onlyLoadOneNode: false, skipJournaledProperties, shouldPassLineNumberInfo, xamlLineInfo, xamlLineInfoConsumer, stack, styleConnector);

xamlObjectWriter.Close();

return xamlObjectWriter.Result;

}

catch (Exception ex)

{

if (CriticalExceptions.IsCriticalException(ex) || !XamlReader.ShouldReWrapException(ex, baseUri))

{

throw;

}

XamlReader.RewrapException(ex, xamlLineInfo, baseUri);

return null;

}

}

internal static void TransformNodes(System.Xaml.XamlReader xamlReader, XamlObjectWriter xamlWriter, bool onlyLoadOneNode, bool skipJournaledProperties, bool shouldPassLineNumberInfo, IXamlLineInfo xamlLineInfo, IXamlLineInfoConsumer xamlLineInfoConsumer, MS.Internal.Xaml.Context.XamlContextStack<WpfXamlFrame> stack, IStyleConnector styleConnector)

{

while (xamlReader.Read())

{

if (shouldPassLineNumberInfo && xamlLineInfo.LineNumber != 0)

{

xamlLineInfoConsumer.SetLineInfo(xamlLineInfo.LineNumber, xamlLineInfo.LinePosition);

}

switch (xamlReader.NodeType)

{

case System.Xaml.XamlNodeType.NamespaceDeclaration:

xamlWriter.WriteNode(xamlReader);

if (stack.Depth == 0 || stack.CurrentFrame.Type != null)

{

stack.PushScope();

for (WpfXamlFrame wpfXamlFrame = stack.CurrentFrame; wpfXamlFrame != null; wpfXamlFrame = (WpfXamlFrame)wpfXamlFrame.Previous)

{

if (wpfXamlFrame.XmlnsDictionary != null)

{

stack.CurrentFrame.XmlnsDictionary = new XmlnsDictionary(wpfXamlFrame.XmlnsDictionary);

break;

}

}

if (stack.CurrentFrame.XmlnsDictionary == null)

{

stack.CurrentFrame.XmlnsDictionary = new XmlnsDictionary();

}

}

stack.CurrentFrame.XmlnsDictionary.Add(xamlReader.Namespace.Prefix, xamlReader.Namespace.Namespace);

break;

case System.Xaml.XamlNodeType.StartObject:

WriteStartObject(xamlReader, xamlWriter, stack);

break;

case System.Xaml.XamlNodeType.GetObject:

xamlWriter.WriteNode(xamlReader);

if (stack.CurrentFrame.Type != null)

{

stack.PushScope();

}

stack.CurrentFrame.Type = stack.PreviousFrame.Property.Type;

break;

case System.Xaml.XamlNodeType.EndObject:

xamlWriter.WriteNode(xamlReader);

if (stack.CurrentFrame.FreezeFreezable && xamlWriter.Result is Freezable freezable && freezable.CanFreeze)

{

freezable.Freeze();

}

if (xamlWriter.Result is DependencyObject dependencyObject && stack.CurrentFrame.XmlSpace.HasValue)

{

XmlAttributeProperties.SetXmlSpace(dependencyObject, stack.CurrentFrame.XmlSpace.Value ? "default" : "preserve");

}

stack.PopScope();

break;

case System.Xaml.XamlNodeType.StartMember:

{

if ((!xamlReader.Member.IsDirective || !(xamlReader.Member == XamlReaderHelper.Freeze)) && xamlReader.Member != XmlSpace.Value && xamlReader.Member != XamlLanguage.Space)

{

xamlWriter.WriteNode(xamlReader);

}

stack.CurrentFrame.Property = xamlReader.Member;

if (!skipJournaledProperties || stack.CurrentFrame.Property.IsDirective)

{

break;

}

WpfXamlMember wpfXamlMember = stack.CurrentFrame.Property as WpfXamlMember;

if (!(wpfXamlMember != null))

{

break;

}

DependencyProperty dependencyProperty = wpfXamlMember.DependencyProperty;

if (dependencyProperty == null || !(dependencyProperty.GetMetadata(stack.CurrentFrame.Type.UnderlyingType) is FrameworkPropertyMetadata frameworkPropertyMetadata) || !frameworkPropertyMetadata.Journal)

{

break;

}

int num = 1;

while (xamlReader.Read())

{

switch (xamlReader.NodeType)

{

case System.Xaml.XamlNodeType.StartMember:

num++;

break;

case System.Xaml.XamlNodeType.StartObject:

{

XamlType type = xamlReader.Type;

XamlType xamlType = type.SchemaContext.GetXamlType(typeof(BindingBase));

XamlType xamlType2 = type.SchemaContext.GetXamlType(typeof(DynamicResourceExtension));

if (num == 1 && (type.CanAssignTo(xamlType) || type.CanAssignTo(xamlType2)))

{

num = 0;

WriteStartObject(xamlReader, xamlWriter, stack);

}

break;

}

case System.Xaml.XamlNodeType.EndMember:

num--;

if (num == 0)

{

xamlWriter.WriteNode(xamlReader);

stack.CurrentFrame.Property = null;

}

break;

case System.Xaml.XamlNodeType.Value:

if (xamlReader.Value is DynamicResourceExtension)

{

WriteValue(xamlReader, xamlWriter, stack, styleConnector);

}

break;

}

if (num == 0)

{

break;

}

}

break;

}

case System.Xaml.XamlNodeType.EndMember:

{

WpfXamlFrame currentFrame = stack.CurrentFrame;

XamlMember property = currentFrame.Property;

if ((!property.IsDirective || !(property == XamlReaderHelper.Freeze)) && property != XmlSpace.Value && property != XamlLanguage.Space)

{

xamlWriter.WriteNode(xamlReader);

}

currentFrame.Property = null;

break;

}

case System.Xaml.XamlNodeType.Value:

WriteValue(xamlReader, xamlWriter, stack, styleConnector);

break;

default:

xamlWriter.WriteNode(xamlReader);

break;

}

if (onlyLoadOneNode)

{

break;

}

}

}

private static void WriteValue(System.Xaml.XamlReader xamlReader, XamlObjectWriter xamlWriter, MS.Internal.Xaml.Context.XamlContextStack<WpfXamlFrame> stack, IStyleConnector styleConnector)

{

if (stack.CurrentFrame.Property.IsDirective && stack.CurrentFrame.Property == XamlLanguage.Shared && bool.TryParse(xamlReader.Value as string, out var result) && !result && !(xamlReader is Baml2006Reader))

{

throw new XamlParseException(SR.Get("SharedAttributeInLooseXaml"));

}

if (stack.CurrentFrame.Property.IsDirective && stack.CurrentFrame.Property == XamlReaderHelper.Freeze)

{

bool flag = Convert.ToBoolean(xamlReader.Value, TypeConverterHelper.InvariantEnglishUS);

stack.CurrentFrame.FreezeFreezable = flag;

if (xamlReader is Baml2006Reader baml2006Reader)

{

baml2006Reader.FreezeFreezables = flag;

}

}

else if (stack.CurrentFrame.Property == XmlSpace.Value || stack.CurrentFrame.Property == XamlLanguage.Space)

{

if (typeof(DependencyObject).IsAssignableFrom(stack.CurrentFrame.Type.UnderlyingType))

{

stack.CurrentFrame.XmlSpace = (string)xamlReader.Value == "default";

}

}

else

{

if (styleConnector != null && stack.CurrentFrame.Instance != null && stack.CurrentFrame.Property == XamlLanguage.ConnectionId && typeof(Style).IsAssignableFrom(stack.CurrentFrame.Type.UnderlyingType))

{

styleConnector.Connect((int)xamlReader.Value, stack.CurrentFrame.Instance);

}

xamlWriter.WriteNode(xamlReader);

}

}

private static void WriteStartObject(System.Xaml.XamlReader xamlReader, XamlObjectWriter xamlWriter, MS.Internal.Xaml.Context.XamlContextStack<WpfXamlFrame> stack)

{

xamlWriter.WriteNode(xamlReader);

if (stack.Depth != 0 && stack.CurrentFrame.Type == null)

{

stack.CurrentFrame.Type = xamlReader.Type;

return;

}

stack.PushScope();

stack.CurrentFrame.Type = xamlReader.Type;

if (stack.PreviousFrame.FreezeFreezable)

{

stack.CurrentFrame.FreezeFreezable = true;

}

}

**Приложение Г**

**Листинг App.xaml.cs**

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Linq;

using System.Threading.Tasks;

using System.Windows;

namespace ChildPlaygroundTracker

{

/// <summary>

/// Логика взаимодействия для App.xaml

/// </summary>

public partial class App : Application

{

}

}