using Microsoft.Win32;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.IO;

using System.Linq;

using System.Runtime.CompilerServices;

using System.Security.Permissions;

using System.Text;

using System.Threading.Tasks;

using System.Xml.Linq;

using VersOne.Epub;

using System.Windows.Media;

using static DevExpress.Utils.HashCodeHelper.Primitives;

using System.Windows.Media.Imaging;

using System.Drawing;

using VersOne.Epub.Schema;

using VersOne.Epub.Options;

namespace eBook\_Reader.Model {

public class Book : INotifyPropertyChanged {

private EpubBook m\_epubBook;

private String m\_bookPath;

private String m\_newBookPath;

private Byte[] m\_coverImage;

private String m\_title;

private String m\_author;

private Boolean m\_isFavorite;

public Book(String bookPath) {

// This object lets majority of invalid files in our application

// We need it because there are many invalid epub files in internet

EpubReaderOptions options = GetReaderOptions();

// Wrap epub file in 'VersOne.Epub.EpubBook asyncronously'

m\_epubBook = EpubReader.ReadBookAsync(bookPath, options).Result;

m\_bookPath = bookPath;

m\_coverImage = m\_epubBook.CoverImage;

m\_title = m\_epubBook.Title;

m\_author = m\_epubBook.Author;

m\_newBookPath = Path.Combine(Properties.LibrarySettings.Default.LibraryPath, Path.GetFileName(m\_bookPath));

}

private EpubReaderOptions GetReaderOptions() {

EpubReaderOptions options = new EpubReaderOptions() {

PackageReaderOptions = new PackageReaderOptions() {

IgnoreMissingToc = true,

SkipInvalidManifestItems = true,

},

Epub2NcxReaderOptions = new Epub2NcxReaderOptions() {

IgnoreMissingContentForNavigationPoints = true

},

XmlReaderOptions = new XmlReaderOptions() {

SkipXmlHeaders = true,

}

};

options.ContentReaderOptions.ContentFileMissing += (sender, e) => {

e.SuppressException = true;

};

return options;

}

public EpubBook EBook {

get => m\_epubBook;

}

public String Title {

get { return m\_title; }

}

public String Author {

get { return m\_author; }

}

public Byte[] CoverImage {

get { return m\_coverImage; }

}

public String BookPath {

get { return m\_bookPath; }

set {

m\_bookPath = value;

OnPropertyChanged("BookPath");

}

}

public String NewBookPath {

get { return m\_newBookPath; }

set {

m\_newBookPath = value;

OnPropertyChanged("NewBookPath");

}

}

public Boolean IsFavorite {

get => m\_isFavorite;

set {

m\_isFavorite = value;

OnPropertyChanged("IsFavorite");

}

}

public event PropertyChangedEventHandler? PropertyChanged;

public void OnPropertyChanged([CallerMemberName] String prop = "") {

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(prop));

}

}

}

using System;

using System.IO;

using System.Windows;

using System.Windows.Forms;

using System.Xml.Linq;

using eBook\_Reader.Model;

using eBook\_Reader.Stores;

using eBook\_Reader.ViewModel;

namespace eBook\_Reader.Commands.ManageLibrary;

public class AddBookCommand : CommandBase

{ \*

\* Class: AddBookCommand

\*

\* Need for adding new books in chosen directory,

\* new item in 'AllbooksViewModel.BookList'

\* and new element 'in BookList.xml'

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

private readonly AllBooksViewModel m\_viewModel;

public AddBookCommand(AllBooksViewModel viewModel) {

m\_viewModel = viewModel;

}

[STAThread]

public override void Execute(object? parameter) {

String fileName = "";

String sourceFilePath = GetFilePathFromDialog(ref fileName);

// Copy the file in new directory and add element in 'BookList.xml'

if (sourceFilePath != "") {

string libraryPath = Properties.LibrarySettings.Default.LibraryPath;

try {

File.Copy(sourceFilePath, Path.Combine(libraryPath, fileName), true);

Book book = new Book(Path.Combine(libraryPath, fileName));

m\_viewModel.BookList.Add(book);

AddToXML(book, m\_viewModel.LibraryPath);

}

catch (AggregateException) {

File.Delete(Path.Combine(libraryPath, fileName));

System.Windows.MessageBox.Show("Something wrong with file", "Error", MessageBoxButton.OK, MessageBoxImage.None);

}

}

return;

}

private String GetFilePathFromDialog(ref String fileName) {

OpenFileDialog openFileDialog = new OpenFileDialog();

String sourceFilePath = "";

try {

openFileDialog.Filter = "EPUB Files(\*.epub)|\*.epub|All files (\*.\*)|\*.\*";

openFileDialog.CheckFileExists = true;

openFileDialog.CheckPathExists = true;

if(openFileDialog.ShowDialog() == DialogResult.OK) {

sourceFilePath = openFileDialog.FileName;

fileName = Path.GetFileName(sourceFilePath);

}

} catch(ArgumentException) {

sourceFilePath = "";

System.Windows.MessageBox.Show("Invalid file format", "Error", MessageBoxButton.OK, MessageBoxImage.None);

}

return sourceFilePath;

}

// Add a line in xml file with attributes:

// 'Name', 'IsFavorite', 'LastOpeningTime', 'progress'

private static void AddToXML(Book book, String? path = null) {

if(path == null || path == "Library")

path = Path.Combine(Environment.CurrentDirectory, "BookList.xml");

XDocument? xdoc = XDocument.Load(path);

XElement? root = xdoc?.Root;

XElement bookElement = new XElement("book");

XAttribute bookNameAttribute = new XAttribute("Name", book.BookPath);

XAttribute bookIsFavoriteAttribute = new XAttribute("IsFavorite", false);

XAttribute bookLastOpeningTime = new XAttribute("LastOpeningTime", new DateTime(1, 1, 1, 1, 1, 1).ToString());

XAttribute bookProgress = new XAttribute("progress", "");

bookElement.Add(bookNameAttribute, bookIsFavoriteAttribute, bookLastOpeningTime, bookProgress);

root?.Add(bookElement);

xdoc?.Save(path);

}

}

using eBook\_Reader.ViewModel;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using WinForms = System.Windows.Forms;

namespace eBook\_Reader.Commands.ManageLibrary

{

class ChangeLibraryPathCommand : CommandBase

{

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Class: ChangeLibraryPathCommand

\*

\* Command for changing library directory

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

private readonly SettingsViewModel m\_settingsViewModel;

public ChangeLibraryPathCommand(SettingsViewModel settingsViewModel) {

m\_settingsViewModel = settingsViewModel;

}

// The method gets a string of new directory path from 'FolderBrowserDialog'

// and sets its value to settings file member

public override void Execute(object? parameter) {

WinForms.FolderBrowserDialog dialog = new WinForms.FolderBrowserDialog();

dialog.InitialDirectory = "C:\\";

WinForms.DialogResult result = dialog.ShowDialog();

if (result == WinForms.DialogResult.OK) {

Properties.LibrarySettings.Default.LibraryPath = dialog.SelectedPath;

m\_settingsViewModel.LibraryPath = dialog.SelectedPath;

Properties.LibrarySettings.Default.Save();

}

}

}

}using eBook\_Reader.Model;

using eBook\_Reader.ViewModel;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Xml;

using System.Xml.Linq;

namespace eBook\_Reader.Commands.ManageLibrary

{

internal class DeleteBookCommand : CommandBase

{

private readonly AllBooksViewModel? m\_allBooksViewModel;

public DeleteBookCommand(AllBooksViewModel? allBooksViewModel)

{

m\_allBooksViewModel = allBooksViewModel;

}

public override void Execute(object? parameter)

{

if (m\_allBooksViewModel!.SelectedBook != null) {

m\_allBooksViewModel.BookList.Remove(m\_allBooksViewModel.SelectedBook);

File.Delete(m\_allBooksViewModel.SelectedBook.NewBookPath);

DeleteFromXML(m\_allBooksViewModel.SelectedBook);

if (m\_allBooksViewModel.SelectedBook.BookPath == m\_allBooksViewModel.LastOpenedBook!.BookPath) {

m\_allBooksViewModel.ContinueReadingVisibility = "Hidden";

}

}

}

// Remove selected book by LINQ to XML

private void DeleteFromXML(Book book)

{

XDocument xDoc = XDocument.Load("BookList.xml");

xDoc?.Descendants()

.Where(e => e.Name == "book")?

.FirstOrDefault(b => b.Attribute("Name")?.Value.Replace('\\', '/') == book?.BookPath.Replace("\\", "/"))?

.Remove();

xDoc?.Save("BookList.xml");

}

}

}using eBook\_Reader.Model;

using eBook\_Reader.ViewModel;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Xml.Linq;

namespace eBook\_Reader.Commands.ManageLibrary

{

public class EmptyLibraryCommand : CommandBase

{

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Class: EmptyLibraryCommand

\*

\* Command for deleting book: it removes chosen element

\* from 'AllBooksViewModel.BookList' and from

\* 'BookList.xml'

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

private readonly AllBooksViewModel m\_allBooksViewModel;

private readonly Boolean m\_defaultSure;

private readonly String? m\_path;

public EmptyLibraryCommand(AllBooksViewModel allBooksViewModel, Boolean defaultSure = false, String? path = null) {

m\_allBooksViewModel = allBooksViewModel;

m\_defaultSure = defaultSure;

m\_path = path;

}

public override void Execute(object? parameter) {

DialogResult dialogResult;

if(!m\_defaultSure) {

dialogResult = MessageBox.Show("Are you sure?",

"Empty library", MessageBoxButtons.YesNo);

} else {

dialogResult = DialogResult.Yes;

}

switch (dialogResult) {

case DialogResult.Yes:

// Delete all elements in 'BookList.xml' with the help of a loop

foreach (var book in m\_allBooksViewModel.BookList) {

File.Delete(book.BookPath);

DeleteFromXML(book, m\_path);

}

m\_allBooksViewModel.BookList.Clear();

break;

case DialogResult.No:

break;

}

}

// Remove selected book by LINQ to XML

private void DeleteFromXML(Book book, String? path = null)

{

XDocument xDoc;

if(path == null)

xDoc = XDocument.Load("BookList.xml");

else

xDoc = XDocument.Load(path);

xDoc?.Descendants()

.Where(e => e.Name == "book")?

.FirstOrDefault(b => b.Attribute("Name")?.Value.Replace('\\', '/') == book?.BookPath.Replace("\\", "/"))?

.Remove();

xDoc?.Save("BookList.xml");

}

}

}using eBook\_Reader.View;

using eBook\_Reader.ViewModel;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Xml.Linq;

namespace eBook\_Reader.Commands.ManageLibrary

{

internal class MarkFavoriteCommand : CommandBase

{

private AllBooksViewModel m\_allBooksViewModel;

public MarkFavoriteCommand(AllBooksViewModel allBooksViewModel)

{

m\_allBooksViewModel = allBooksViewModel;

}

public override void Execute(object? parameter)

{

string path = Path.Combine(Environment.CurrentDirectory, "BookList.xml");

XElement? xElement = XElement.Load(path);

foreach (var Xbook in xElement.DescendantsAndSelf("book"))

{

if (Xbook.Attribute("Name")?.Value.Replace('\\', '/') == m\_allBooksViewModel.SelectedBook!.BookPath.Replace("\\", "/"))

{

Xbook.SetAttributeValue("IsFavorite", "true");

xElement.Save(path);

m\_allBooksViewModel.SelectedBook.IsFavorite = true;

}

}

}

}

}

using eBook\_Reader.ViewModel;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Xml.Linq;

namespace eBook\_Reader.Commands.ManageLibrary

{

internal class RemoveFavoriteMarkCommand : CommandBase

{

private AllBooksViewModel m\_allBooksViewModel;

private FavoriteBooksViewModel? m\_favoriteBooksViewModel;

public RemoveFavoriteMarkCommand(AllBooksViewModel allBooksViewModel,

FavoriteBooksViewModel? favoriteBooksViewModel = null)

{

m\_allBooksViewModel = allBooksViewModel;

m\_favoriteBooksViewModel = favoriteBooksViewModel;

}

public override void Execute(object? parameter) {

String path = Path.Combine(Environment.CurrentDirectory, "BookList.xml");

XElement? xElement = XElement.Load(path);

if (m\_favoriteBooksViewModel == null) {

foreach (var Xbook in xElement.DescendantsAndSelf("book")) {

if (Xbook.Attribute("Name")?.Value.Replace('\\', '/') == m\_allBooksViewModel.SelectedBook!.BookPath.Replace("\\", "/")) {

Xbook.SetAttributeValue("IsFavorite", "false");

xElement.Save(path);

m\_allBooksViewModel.SelectedBook.IsFavorite = false;

}

}

}

else {

foreach (var Xbook in xElement.DescendantsAndSelf("book")) {

if (Xbook.Attribute("Name")?.Value.Replace('\\', '/') == m\_favoriteBooksViewModel.SelectedBook.BookPath.Replace("\\", "/")) {

Xbook.SetAttributeValue("IsFavorite", "false");

xElement.Save(path);

m\_allBooksViewModel.SelectedBook!.IsFavorite = false;

}

}

foreach (var book in m\_allBooksViewModel.BookList) {

if (m\_favoriteBooksViewModel.SelectedBook.BookPath == book.BookPath) {

m\_favoriteBooksViewModel.SelectedBook.IsFavorite = false;

book.IsFavorite = false;

}

}

}

}

}

}using eBook\_Reader.Model;

using eBook\_Reader.ViewModel;

using Microsoft.VisualBasic;

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using static System.Reflection.Metadata.BlobBuilder;

namespace eBook\_Reader.Commands.ManageLibrary {

class SortCommand<T> : CommandBase

where T : BooksViewModel {

private readonly T m\_viewModel;

public SortCommand(T TViewModel) {

m\_viewModel = TViewModel;

}

public override void Execute(object? parameter) {

String? SelectedSortProperty = m\_viewModel.SelectedSortParameter.Name;

ObservableCollection<Book> books = m\_viewModel.BookList;

List<Book> tempList;

switch (SelectedSortProperty) {

case "TitleUp": {

tempList = books.OrderBy(book => book.Title).ToList();

BackToObservableCollection(tempList, ref books);

break;

}

case "TitleDown": {

tempList = books.OrderByDescending(book => book.Title).ToList();

BackToObservableCollection(tempList, ref books);

break;

}

case "AuthorUp": {

tempList = books.OrderBy(book => book.Author).ToList();

BackToObservableCollection(tempList, ref books);

break;

}

case "AuthorDown": {

tempList = books.OrderByDescending(book => book.Author).ToList();

using eBook\_Reader.Model;

using eBook\_Reader.ViewModel;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Xml;

using System.Xml.Linq;

using System.Xml.Serialization;

using System.Xml.XPath;

namespace eBook\_Reader.Utils {

internal class ProgressSerializer {

internal static void SerializeProgress(SerializableProgress serializableProgress) {

String path = Path.Combine(Environment.CurrentDirectory, "BookProgress.xml");

XDocument? xdoc = XDocument.Load(path);

using(XmlWriter writer = xdoc.Root.CreateWriter()) {

new XmlSerializer(serializableProgress.GetType()).Serialize(writer, serializableProgress);

}

}

internal static SerializableProgress DeserializeProgress(ReadBookViewModel readBookViewModel) {

XmlSerializer xmlSerializer = new XmlSerializer(typeof(SerializableProgress));

String fileName = "BookProgress.xml";

String path = Path.Combine(Environment.CurrentDirectory, fileName);

XElement xElement = XElement.Load(path);

XElement? book = (from xbook in xElement.DescendantsAndSelf("SerializableProgress")

where (xbook.Attribute("BookName")?.Value.Replace('\\', '/') == readBookViewModel.SelectedBook.BookPath.Replace("\\", "/"))

select xbook).FirstOrDefault();

SerializableProgress? serializableProgress;

if (book != null) {

serializableProgress = xmlSerializer.Deserialize(book.CreateReader()) as SerializableProgress;

return serializableProgress;

}

return null;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Xml.Serialization;

namespace eBook\_Reader.Utils {

[Serializable]

public class SerializableProgress {

[XmlElement("name")]

public String BookName { get; set; }

[XmlElement("point")]

public String PointText { get; set; }

public SerializableProgress() {

BookName = "";

PointText = "";

}

public SerializableProgress(String bookName, String pointText) {

BookName = bookName;

PointText = pointText;

}

}

}

}

}

}

BackToObservableCollection(tempList, ref books);

break;

}

default:

return;

}

}

private void BackToObservableCollection(List<Book> tempList, ref ObservableCollection<Book> books) {

for (int i = 0; i < tempList.Count; i++) {

books.Move(books.IndexOf(tempList[i]), i);

}

}

}

}

using eBook\_Reader.Model;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Xml.Linq;

using System.Xml.Serialization;

namespace eBook\_Reader.Utils {

public class Serializer {

public static void Serialize(SerializableProgress progress) {

String path = Path.Combine(Environment.CurrentDirectory, "BookProgress.xml");

XDocument? xdoc = XDocument.Load(path);

XElement? root = xdoc?.Root;

XElement bookElement = new XElement("book");

XAttribute bookNameAttribute = new XAttribute("name", progress.BookName);

XAttribute bookTextPointAttribute = new XAttribute("textPoint", progress.PointText);

bookElement.Add(bookNameAttribute, bookTextPointAttribute);

root?.Add(bookElement);

xdoc.Save(path);

}

public static String? Deserialize() {

XmlSerializer xSerializer = new XmlSerializer(typeof(SerializableProgress));

using(FileStream fs = new FileStream("BookProgress.xml", FileMode.OpenOrCreate)) {

return (String?) xSerializer.Deserialize(fs);

}

}

}

}

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Linq;

using System.Threading.Tasks;

using System.Windows;

using eBook\_Reader.Stores;

using eBook\_Reader.View;

using eBook\_Reader.ViewModel;

namespace eBook\_Reader {

/// <summary>

/// Interaction logic for App.xaml

/// </summary>

public partial class App : Application {

protected override void OnStartup(StartupEventArgs e) {

NavigationStore m\_navigationStore = new NavigationStore();

MenuNavigationStore m\_menuNavigationStore = new MenuNavigationStore();

AllBooksViewModel allBooksViewModelRef = new AllBooksViewModel(m\_navigationStore, m\_menuNavigationStore);

m\_menuNavigationStore.CurrentMenuViewModel = allBooksViewModelRef;

m\_navigationStore.CurrentViewModel = new MainMenuViewModel(m\_navigationStore, m\_menuNavigationStore, allBooksViewModelRef);

MainWindow = new MainWindow() {

DataContext = new MainViewModel(m\_navigationStore)

};

MainWindow.Show();

base.OnStartup(e);

}

}

}