Tristan Izlar

COP2362

TUTORIAL 5-2: Decoupling Application logic and Handling Events

I worked alone.

A picture containing text, monitor, screenshot, indoor

Description automatically generated

Graphical user interface

Description automatically generated

Auditor.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Windows.UI.Popups;

using Windows.Storage.Pickers;

using Windows.Storage;

using Windows.Data.Xml.Dom;

using DataTypes;

namespace AuditService

{

public class Auditor

{

public delegate void AuditingCompleteDelegate(string message);

public event AuditingCompleteDelegate AuditProcessingComplete;

public void AuditOrder(Order order)

{

this.doAuditing(order);

}

private async void doAuditing(Order order)

{

List<OrderItem> ageRestrictedItems = findAgeRestrictedItems(order);

if (ageRestrictedItems.Count > 0)

{

try

{

StorageFile file = await ApplicationData.Current.LocalFolder.CreateFileAsync($"audit-{order.OrderID}.xml");

if (file != null)

{

XmlDocument doc = new XmlDocument();

XmlElement root = doc.CreateElement("Order");

root.SetAttribute("ID", order.OrderID.ToString());

root.SetAttribute("Date", order.Date.ToString());

foreach (OrderItem orderItem in ageRestrictedItems)

{

XmlElement itemElement = doc.CreateElement("Item");

itemElement.SetAttribute("Product", orderItem.Item.Name);

itemElement.SetAttribute("Description", orderItem.Item.Description);

root.AppendChild(itemElement);

}

doc.AppendChild(root);

await doc.SaveToFileAsync(file);

}

else

{

MessageDialog dlg = new MessageDialog($"Unable to save to file: {file.DisplayName}", "Not saved");

dlg.ShowAsync();

}

}

catch (Exception ex)

{

MessageDialog dlg = new MessageDialog(ex.Message, "Exception");

dlg.ShowAsync();

}

finally

{

if(this.AuditProcessingComplete != null)

{

this.AuditProcessingComplete($"Audit record written for Order {order.OrderID}");

}

}

}

}

private List<OrderItem> findAgeRestrictedItems(Order order)

{

return order.Items.FindAll(o => o.Item.AgeRestricted == true);

}

}

}

ChechoutController.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using DataTypes;

namespace CheckoutService

{

public class CheckoutController

{

public delegate void CheckoutDelegate(Order order);

public CheckoutDelegate CheckoutProcessing = null;

private bool requestPayment()

{

return true;

}

public void StartCheckoutProcessing(Order order)

{

if(this.requestPayment())

{

if(this.CheckoutProcessing!=null)

{

this.CheckoutProcessing(order);

}

}

}

}

}

Order.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DataTypes

{

public class Order

{

public Guid OrderID { get; set; }

public DateTime Date { get; set; }

public decimal TotalValue { get; set; }

public List<OrderItem> Items { get; set; }

}

}

OrderItem.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DataTypes

{

public class OrderItem

{

public Product Item { get; set; }

public int Quantity { get; set; }

}

}

Product.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DataTypes

{

public class Product

{

public string ProductID { get; set; }

public string Name { get; set; }

public string Description { get; set; }

public decimal Price { get; set; }

public bool AgeRestricted { get; set; }

}

}

ProductDatasource.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DataTypes

{

public class ProductsDataSource

{

public List<Product> Products { get; set; }

public ProductsDataSource()

{

Products = new List<Product>

{

new Product { ProductID="P1", Name="Rope", Description="Best Italian hemp, 40ft", AgeRestricted=false, Price=28.00M },

new Product { ProductID="P2", Name="Wood", Description="Pine, 4\' x 2\' x 18\'", AgeRestricted=false, Price=12.20M },

new Product { ProductID="P3", Name="Screwdriver", Description="Crossheaded", AgeRestricted=false, Price=4.99M },

new Product { ProductID="P4", Name="Power Drill", Description="1800 RPM hammer drill", AgeRestricted=true, Price=75.50M },

new Product { ProductID="P5", Name="Hammer", Description="24oz heavy-duty claw hammer", AgeRestricted=false, Price=18.35M },

new Product { ProductID="P6", Name="Power Saw", Description="Rotary action, high powered", AgeRestricted=true, Price=88.00M },

new Product { ProductID="P7", Name="Nails", Description="2\" masonry nails", AgeRestricted=false, Price=5.00M },

new Product { ProductID="P8", Name="Saw", Description="Fine-toothed fretsaw", AgeRestricted=false, Price=3.20M },

};

}

}

}

Shipper.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Windows.Storage;

using Windows.Storage.Streams;

using Windows.UI.Popups;

using DataTypes;

namespace DeliveryService

{

public class Shipper

{

public delegate void ShippingCompleteDelegate(string message);

public event ShippingCompleteDelegate ShipProcessingComplete;

public void ShipOrder(Order order)

{

this.doShipping(order);

}

private async void doShipping(Order order)

{

try

{

StorageFile file = await ApplicationData.Current.LocalFolder.CreateFileAsync($"dispatch-{order.OrderID}.txt");

if (file != null)

{

string dispatchNote = $"Order Summary: \r\nOrder ID: {order.OrderID}\r\nOrder Total: {order.TotalValue:C}";

var stream = await file.OpenAsync(FileAccessMode.ReadWrite);

var writeStream = stream.GetOutputStreamAt(0);

DataWriter writer = new DataWriter(writeStream);

writer.WriteString(dispatchNote);

await writer.StoreAsync();

await writeStream.FlushAsync();

writeStream.Dispose();

}

else

{

MessageDialog dlg = new MessageDialog($"Unable to save to file: {file.DisplayName}", "Not saved");

dlg.ShowAsync();

}

}

catch (Exception ex)

{

MessageDialog dlg = new MessageDialog(ex.Message, "Exception");

dlg.ShowAsync();

}

finally

{

if(this.ShipProcessingComplete != null)

{

this.ShipProcessingComplete($"Dispatch note generated for Order {order.OrderID}");

}

}

}

}

}