

```
1
2
3 using System;
4 using System.Collections.Generic;
5 using System.Linq;
6 using System.Text;
7 using System.Threading.Tasks;
8 using System.Windows;
9 using System.Windows.Controls;
10 using System.Windows.Data;
11 using System.Windows.Documents;
12 using System.Windows.Input;
13 using System.Windows.Media;
14 using System.Windows.Media.Imaging;
15 using System.Windows.Navigation;
16 using System.Windows.Shapes;
17
18 namespace Motoring
19 {
20     /// <summary>
21     /// Interaction logic for PageRunning.xaml
22     /// </summary>
23     public partial class PageRunning : Page
24     {
25         //-- variables to hold period values selected in the combos
26         //-- Placing them here makes them global in the module
27         //-- This is a different approach to previous used
28         private string insurancePeriod = "Annual";    //-- Initialised Annual
29         private string fuelPeriod = "Annual";        //-- Ditto
30         private string servicingPeriod = "Annual";   //-- Ditto
31         private string roadTaxPeriod = "Annual";     //-- Ditto
32         //-- Numeric values
33         private int insurance = 0;                   //-- Initialised to 0
34         private int fuel = 0;                         //-- Ditto
35         private int servicing = 0;                    //-- Ditto
36         private int roadTax = 0;                      //-- Ditto
37
38         //-- Combo items list of strings
39         List<string> costPeriods = new List<string>();
40
41         //-- Object to manage data brought forward
42         CostSummary summaryBroughtForward = new CostSummary();
43
44         public PageRunning(CostSummary summaryPassed)
45         {
46             InitializeComponent();
47             summaryBroughtForward = summaryPassed;
48             costPeriods = Periods();
49
50             //-- Combo data sources (maybe redundant)
51             InsurancePeriodCombo.ItemsSource = Periods();
52             FuelPeriodCombo.ItemsSource = Periods();
53             ServicingPeriodCombo.ItemsSource = Periods();
54             RoadTaxPeriodCombo.ItemsSource = Periods();
55         }
56     }
```

```
57     private void SummaryPageButton_OnClick(object sender, RoutedEventArgs e)
58     {
59         CostSummary summary = new CostSummary();
60
61         summary = summaryBroughtForward;
62
63
64         //-- Run the method to get the values in the form controls
65         HarvestValues();
66
67         //-- object to hold the RunningCost data
68         //-- Initialise the object with the current values held in the module-wide variables
69         RunningCost runningCost = new RunningCost
70         {
71             Insurance = insurance,
72             InsurancePeriod = insurancePeriod,
73             Fuel = fuel,
74             FuelPeriod = fuelPeriod,
75             Servicing = servicing,
76             ServicingPeriod = servicingPeriod,
77             RoadTax = roadTax,
78             RoadTaxPeriod = roadTaxPeriod
79         };
80
81         summary.RunningCost = runningCost;
82
83         var pageSummary = new PageSummary(summary);
84         this.NavigationService.Navigate(pageSummary);
85     }
86
87     // -- Combo box data population
88     List<string> Periods()
89     {
90         List<string> myList = new List<string>
91         {
92             "Annual",
93             "Monthly",
94             "Weekly"
95         };
96
97         return myList;
98     }
99
100
101     //-- Combo Box Load events
102     private void InsurancePeriodCombo_OnLoaded(object sender, RoutedEventArgs e)
103     {
104         var combo = sender as ComboBox;
105         combo.ItemsSource = costPeriods;
106         combo.SelectedIndex = 0;
107     }
108
109     private void FuelPeriodCombo_OnLoaded(object sender, RoutedEventArgs e)
```

```
e)
{
110     var combo = sender as ComboBox;
111     combo.ItemsSource = costPeriods;
112     combo.SelectedIndex = 0;
113 }
114
115
116 private void ServicingPeriodCombo_OnLoaded(object sender,      ↗
    RoutedEventArgs e)
117 {
118     var combo = sender as ComboBox;
119     combo.ItemsSource = costPeriods;
120     combo.SelectedIndex = 0;
121 }
122
123 private void RoadTaxPeriodCombo_OnLoaded(object sender,      ↗
    RoutedEventArgs e)
124 {
125     var combo = sender as ComboBox;
126     combo.ItemsSource = costPeriods;
127     combo.SelectedIndex = 0;
128 }
129
130 private void InsurancePeriodCombo_OnSelectionChanged(object sender,      ↗
    SelectionChangedEventArgs e)
131 {
132     var selectedComboItem = sender as ComboBox;
133     insurancePeriod = selectedComboItem.SelectedItem as string;
134 }
135
136 private void FuelPeriodCombo_OnSelectionChanged(object sender,      ↗
    SelectionChangedEventArgs e)
137 {
138     var selectedComboItem = sender as ComboBox;
139     fuelPeriod = selectedComboItem.SelectedItem as string;
140 }
141
142 private void ServicingPeriodCombo_OnSelectionChanged(object sender,      ↗
    SelectionChangedEventArgs e)
143 {
144     var selectedComboItem = sender as ComboBox;
145     servicingPeriod = selectedComboItem.SelectedItem as string;
146 }
147
148 private void RoadTaxPeriodCombo_OnSelectionChanged(object sender,      ↗
    SelectionChangedEventArgs e)
149 {
150     var selectedComboItem = sender as ComboBox;
151     roadTaxPeriod = selectedComboItem.SelectedItem as string;
152 }
153
154 //-- Harvest numeric values from the Text Boxes
155 private void HarvestValues()
156 {
157     //-- Insurance
158     if (InsuranceCostTextBox.Text != "")
```

```
159     {
160         if (int.TryParse(InsuranceCostTextBox.Text, out int insuranceValue))
161         {
162             insurance = insuranceValue;
163         }
164         else
165         {
166             MessageBox.Show("The insurance cost must be a whole number");
167         }
168     }
169     else
170     {
171         MessageBox.Show("Enter the insurance cost, even if it's 0");
172     }
173
174     //-- Fuel
175     if (FuelCostTextBox.Text != "")
176     {
177         if (int.TryParse(FuelCostTextBox.Text, out int fuelValue))
178         {
179             fuel = fuelValue;
180         }
181         else
182         {
183             MessageBox.Show("The fuel cost must be a whole number");
184         }
185     }
186     else
187     {
188         MessageBox.Show("You must enter a fuel cost, even if it's 0");
189     }
190
191     //-- Servicing
192     if (ServicingCostTextBox.Text != "")
193     {
194         if (int.TryParse(ServicingCostTextBox.Text, out int servicingValue))
195         {
196             servicing = servicingValue;
197         }
198         else
199         {
200             MessageBox.Show("The servicing cost must be a whole number, even if it's 0");
201         }
202     }
203     else
204     {
205         MessageBox.Show("You must enter a servicing cost, even if it's 0");
206     }
207
208     //-- Road tax
209     if (RoadTaxCostTextBox.Text != "")
```

```
210     {
211         if (int.TryParse(RoadTaxCostTextBox.Text, out int roadTaxValue))
212         {
213             roadTax = roadTaxValue;
214         }
215         else
216         {
217             MessageBox.Show("The road tax amount must be a whole number, even if it's 0");
218         }
219     }
220     else
221     {
222         MessageBox.Show("you must enter a Road Tax value, even if it's 0");
223     }
224 }
225 }
226 }
227 }
228 }
```