Vendor: Microsoft

Exam Code: 70-506

Exam Name: TS: Silverlight 4, Development

Version: DEMO

1.You are developing a Silverlight 4 application.
The application defines the following three event handlers. (Line numbers are included for reference only.)

```
01 private void HandleCheck(object sender, RoutedEventArgs e)
02 {
03 MessageBox.Show("Checked")
04 }
05
06 private void HandleUnchecked(object sender, RoutedEventArgs e)
07 {
08 MessageBox.Show("Unchecked")
09 }
10
11 private void HandleThirdState(object sender, RoutedEventArgs e)
12 {
13 MessageBox.Show("Indeterminate")
14 }
```

You need to allow a check box that can be selected, cleared, or set to Indeterminate. You also need to ensure that the event handlers are invoked when the user changes the state of the control.
Which XAML fragment should you use?
A. <CheckBox x:Name="cb2" Content="Three State CheckBox" IsChecked="True" Checked="HandleCheck" Indeterminate="HandleUnchecked" Unchecked="HandleUnchecked" />
B. <CheckBox x:Name="cb2" Content="Three State CheckBox" IsThreeState="True" Checked="HandleCheck" Indeterminate="HandleThirdState" Unchecked="HandleUnchecked" />
C. <CheckBox x:Name="cb2" Content="Three State CheckBox" IsHitTestVisible="True" Checked="HandleCheck" Indeterminate="HandleThirdState" Unchecked="HandleUnchecked" />
D. <CheckBox x:Name="cb2" Content="Three State CheckBox" IsEnabled="True" Checked="HandleCheck" Indeterminate="HandleUnchecked" Unchecked="HandleUnchecked" />
Answer: B

2.You are developing a Silverlight 4 application.
The application contains an XAML page that defines the following Grid control.

```
<Grid Name="gridBody" >
<Grid.RowDefinitions>
<RowDefinition />
<RowDefinition />
</Grid.RowDefinitions>
<TextBlock Text="Employee Info" />
```

<TextBlock Text="Please enter employee info" Grid.Row="1" Height="20"
VerticalAlignment="Top" />
<TextBox x:Name="EmpInfo" Grid.Row="1" Margin="0,25,0,0" TextWrapping="Wrap" />
...
</Grid>
The codebehind file for myPage.xaml contains the following code segment. (Line numbers are included for reference only.)
01 public myPage()
02 {
03 InitializeComponent()
04
05 UserControl control = new MyCustomControl()
06
07 }
You need to replace the contents of the second row of gridBody with a user control of the MyCustomControl type.
Which code segment should you insert at line 06?
A. gridBody.Children.Insert(1, control)
B. gridBody.RowDefinitions.Remove(gridBody.RowDefinitions[1])
gridBody.Children.Insert(1, control)
C. gridBody.Children.Clear()
Grid.SetRow(control, 1)
gridBody.Children.Add(control)
D. List<UIElement> remove = gridBody.Children.Where(c => c is FrameworkElement &&
Grid.GetRow((FrameworkElement)c) == 1).ToList()
foreach (UIElement element in remove)
{
gridBody.Children.Remove(element)
}
Grid.SetRow(control, 1)
gridBody.Children.Add(control)
Answer: D

3.You are developing a Silverlight 4 application.
The application defines the following XAML fragment. (Line numbers are included for reference only.)
01 <ComboBox>
02 <ComboBoxItem Content="Item 1" />
03 <ComboBoxItem Content="Item 2" />
04 <ComboBoxItem Content="Item 3" />
05 </ComboBox>
The codebehind file contains the following code segment. (Line numbers are included for reference only.)
06 void PrintText(object sender, SelectionChangedEventArgs args){

07
08 MessageBox.Show("You selected " + cbi.Content.ToString() + ".")
09 }
You need to ensure that when the user selects an item in a ComboBox control, the content of the item is displayed.
What should you do?
A. Replace the following XAML fragment at line 01.
<ComboBox SelectionChanged="PrintText">
Add the following code segment at line 07.
ComboBoxItem cbi = ((sender as ComboBox).SelectedItem as ComboBoxItem)
B. Replace the following XAML fragment at line 01.
<ComboBox SelectionChanged="PrintText">
Add the following code segment at line 07.
ComboBoxItem cbi = ((sender as ComboBox).SelectedIndex as ComboBoxItem)
C. Replace the following XAML fragment at line 01.
<ComboBox DropDownClosed="PrintText">
Add the following code segment at line 07.
ComboBoxItem cbi = ((sender as ComboBox).SelectedItem as ComboBoxItem)
D. Replace the following XAML fragment at line 01.
<ComboBox DropDownClosed="PrintText">
Add the following code segment at line 07.
ComboBoxItem cbi = ((sender as ComboBox).SelectedIndex as ComboBoxItem)
Answer: A

4.You are developing a Silverlight 4 application.
You have a collection named ColPeople of the List<Person> type. You define the Person class according to the following code segment.
public class Person
{
public string Name {get set }
public string Description { get set }
public string Gender { get set }
public int Age { get set }
public int Weight { get set }
}
You need to bind ColPeople to a ComboBox so that only the Name property is displayed.
Which XAML fragment should you use?
A. <ComboBox DataContext="{Binding ColPeople}" ItemsSource="{Binding ColPeople}" DisplayMemberPath="Name" />
B. <ComboBox DataContext="{Binding Person}" ItemsSource="{Binding Person}" DisplayMemberPath="ColPeople" />
C. <ComboBox DataContext="{Binding ColPeople}" DisplayMemberPath="Name" />
D. <ComboBox DataContext="{Binding Person}" />
Answer: A

5.You are developing a Silverlight 4 application. You define an Invoice object according to the following
code segment.
public class Invoice
{
public int InvoiceId { get set }
public double Amount { get set }
public Supplier Supplier { get set }
public DateTime InvoiceDate { get set }
public DateTime PayDate { get set }
public string InvoiceDescription { get set }
}
You need to display a list of invoices that have the following properties displayed on each line: InvoiceId,
Amount, and InvoiceDate.
Which XAML fragment should you use?
A. <ListBox x:Name="InvoiceListBox">
<StackPanel Orientation="Horizontal">
<TextBlock Text="{Binding Path=InvoiceId}" />
<TextBlock Text="{Binding Path=Amount}" />
<TextBlock Text="{Binding Path=InvoiceDate}" />
</StackPanel>
</ListBox>
B. <ListBox x:Name="InvoiceListBox">
<StackPanel Orientation="Horizontal">
<ListBoxItem>
<TextBlock Text="{Binding Path=InvoiceId}" />
</ListBoxItem>
<ListBoxItem>
<TextBlock Text="{Binding Path=Amount}" />

</ListBoxItem>
<ListBoxItem>
<TextBlock Text="{Binding Path=InvoiceDate}" />
</ListBoxItem>
</StackPanel>
</ListBox>
C. <ListBox x:Name="InvoiceListBox">
<ListBox.Items>
<ItemsPanelTemplate>
<StackPanel Orientation="Horizontal">
<TextBlock Text="{Binding Path=InvoiceId}" />

<TextBlock Text="{Binding Path=Amount}" />
<TextBlock Text="{Binding Path=InvoiceDate}" />
</StackPanel>
</ItemsPanelTemplate>
</ListBox.Items>
</ListBox>
D. <ListBox x:Name="InvoiceListBox">
<ListBox.ItemTemplate>
<DataTemplate>
<StackPanel Orientation="Horizontal">
<TextBlock Text="{Binding Path=InvoiceId}" />
<TextBlock Text="{Binding Path=Amount}" />
<TextBlock Text="{Binding Path=InvoiceDate}" />
</StackPanel>
</DataTemplate>
</ListBox.ItemTemplate>
</ListBox>
Answer: D