Översikt

- Använd XAML för att designa ett användaregränssnitt.
- Knyta controls till information.
- Styla ett användaregränssnitt.
- Skapa egna controls.

Lektion 1: Använd XAML för att designa ett användaregränssnitt

- Introduktion XAML(Extensible Application Markup Language).
- Common Controls.
- Hantera händelser.
- Använda Layout Controls.
- De vanligast Layout Controls.

Introduktion XAML

- XAML ger möjlighet att definiera grafiska element för din .NET Framework applikation.
- Använder sig av XML-element och attribut för att representera controls och dess egenskaper.
- Ett antal verktyg kan användas:
  - Visual Studio.
  - Microsoft Expression Suite.
  - Texteditors.
- När WPF-applikation byggs, kommer motor att konvertera detta till klasser och objekt.

.easec

Introduktion XAML (forts.)

- Hierarki används för att representera föräldrer och barn controls.
- Exempel:

```
Window x:Class="MyNameSpace.MainWindow"
       xmlns
```

.easec

Common Controls

- .NET Framework tillhandahåller en stor samling av controllers som kan användas för att implementera ditt användaregränssnitt.
- Tillgängliga controllers visas i Toolbox.
- Egna controllers kan definieras, user controlls.
- Grid läggs till automatiskt i ett WPF-projekt.

.easec

Common Controls (forts.)

Control	Beskrivning
Button	Visar en knapp som användare kan klicka på för att utföra händelse.
CheckBox	Ger möjlighet att låta användare indikera om ett något skall väljas (checked) eller inte (blankt).
ComboBox	Visar en dropdownruta, med lista av val som användare kan välja ifrån.
Label	Visar statisk text.
ListBox	Visar en skrollbar lista av val som användare kan välja ifrån.
RadioButton	Ger möjlighet för användare att välja mellan flera exklusiva alternativ.
TabControl	Organiserar controls i UI i ett antal sidor, som användare kan använda tabb.
TextBlock	Visar block av text som är read-only.

.eorec

Sätta egenskaper för control

- När du lägger till controls i ett XAML-fönster, kan du definiera egenskaper på olika sätt.
- De flesta controls ger möjlighet åt dig att sätta dessa med attribut.
- Exempel:

```
<Button Content="Jag vill ha kaffe!"
        Margin="150, 130, 150, 130"
        Background="Yellow"
        Foreground="Blue" />
```

.eorec

Sätta egenskaper för control (forts.)

- Att använda attribut, ger inte den flexibilitet som önskas.
- Då kan property element syntax användas.
- Exempel:

```
<Button Content="Gör mig en kopp kaffe!"
        Margin="150, 130, 150, 130">
  <Button.Background>
     <LinearGradientBrush StartPoint="0.5 0.5" EndPoint="1.5 1.5" >
       <Gradient.Stop Color="AliceBlue" Offset="0.5" />
       <Gradient.Stop.Color="Aqua" Offset="0.5" />
     </LinearGradientBrush>
  </Button.Background>
  <Button.Foreground>
        <SolidColorBrush Color="Black" />
  </Button.Foreground>
</Button>
```

.eorec

Sätta egenskaper för control (forts.)

- Många controls inkluderar egenskapen Content.
- I föregående exempel användes attribut för att sätta ett textvärde, kan även användas för att referera till en bild.
- Exempel:

```
<Button Margin="150, 130, 150, 130">
       <Image Source="Images/coffee.jpg" Stretch="Fill" />
</Button>
```

.eorec

Hantera händelser

- När du skapar en WPF-applikation, kommer Visual Studio att skapa en motsvarande .cs-fil.
- I denna fil "prenumererar" du på händelser och definierar logiken.
- När du dubbelklickar t ex på Button (design view), kommer Visual Studio automatiskt skapa event handler.
- Routed Events – när händelse sker, kommer WPF att köra knuten händelse, för de controls som är i fokus.
- Om det inte finns någon tillgänglig, kommer WPF att titta på föräldrer.

.eorec

Hantera händelser (forts.)

- Specificera metod för event handler i XAML:

```
<Button x:Name = "btnMakeCoffee"
               Margin="150, 130, 150, 130"
               Content = "Gör mig en kopp kaffe!"
               Click = "btnMakeCoffee.Click" />
Label x:Name="lblResult"
               Content= ""
               Margin="150, 186, 150, 75" />
```

- Hantera event i code-behind class:

```
private void btnMakeCoffee.Clickobject.sender.RoutedEventArgs e)
{
       lblResult.Content = "Ditt kaffe är på väg.";
}
```

.eorec

Använda Layout Controls

- Relativ positionering är ett av grundkoncepten för WPF.
- Applikationen skall renderas korrekt oavsett hur användaren positionerar eller storleksförändrar fönstret.
- WPF inkluderar ett antal layouter, eller behållare, som ger möjlighet till att ändra position eller storlek på child controls.

.eosec

De vanligast Layout Controls

Control	Beskriving
Canvas	Child controls definierar sin egen layout, genom att specificera koordinater för canvas.
DockPanel	Child controls knyts till DockPanel:s kant.
Grid	Child controls läggs till inuti grid.
StackPanel	Child controls läggs på hög, antingen vertikalt eller horisontellt.
VirtualizingPanel	Child controls läggs på hög, antingen vertikalt eller horisontellt. Men bara de controls som är synliga på skärm skapas.

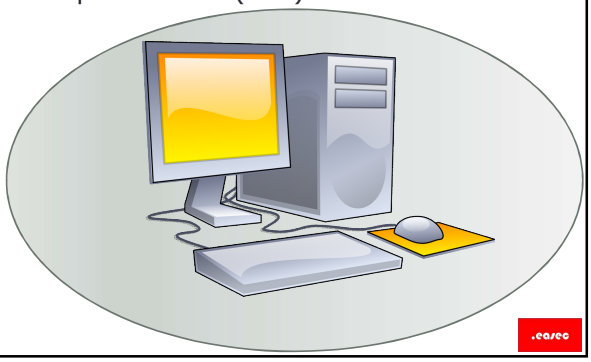
.eosec

Skapa egna controls

- Egna controls kan skapas i WPF.
- Exempelvis om samma kombination av knapp och etikett används på många ställen, är det bättre att skapa en user control
- User control består av en XAML-fil med tillhörande .cs fil

.eosec

Övning Arbeta med Design View för att skapa XAML UI (11a)



Lektion 2: Knyta controls till information

- Introduktion till Data Binding.
- Vägar.
- Knyta controls till information i XAML.
- Knyta controls till information programmässigt.
- Knyta controls till samling.

Introduktion till Data Binding

- Data binding – att ansluta källa för information till ett UI-element.
- Består av tre delar:
  - Binding source, är källan för information.
  - Binding target, element som du vill binda till.
  - Binding object, objekt som knyter källa till mål (target), kan vara converter om olika datatyper.

Vägar

- Vägar:
  - Two way, uppdaterar bägge vägarna.
  - One way, uppdaterar mål om källa uppdaterats.
  - One time, uppdaterar mål när applikation startas, eller DataContext förändras.
  - OneWayToSource, uppdaterar källa när mål förändras.
  - Default.

```
<TextBox Text="{Binding Source={StaticResource coffee1}, Path=Bean,
Mode=TwoWay}" />
```

.easec

Knyta controls till information i XAML

- Du kan knyta controls till information på många olika sätt.
- Om källa inte kommer att ändras under exekvering, kan statisk resurs skapas.

```
<Window x:Class="DataBinding.MainWindow"
        xmlns=http://schemas.microsoft.com/winfx/2006/xaml/presentation
        xmlns:x=http://schemas.microsoft.com/winfx/2006/xaml
        xmlns:loc="clr-namespace:DataBinding"
        Title="Exempel DataBinding" Height="350" Width="525">
  <Window.Resources>
    <loc:Coffee x:Key="coffee1"
                        Name="Fourth Coffee Quencher"
                        Bean="Arabica"
                        CountryOfOrigin="Brazil"
                        Strength="3" />
  </Window.Resources>
  ...
</Window>
```

.easec

Knyta controls till information programmässigt

- I verkligheten är det ovanligt att källinformationen är statisk.
- Vanligtvis inhämtar vi information från databas eller webbtjänst.
- I dessa scenario kan du inte använda statisk resurs, för att representera din information, istället får du använda kod som specificerar informationskälla.

.easec

Knyta controls till information programmässigt -exempel

```
private void mainWindow_Loaded(object sender, RoutedEventArgs e)
{
// Skapa instans för att använda som informationskälla
        Coffee coffee1 = new Coffee();
        coffee1.Name = ”Fourth Coffee Quencher”;
        coffee1.Bean = ”Arabica”;
        Coffee1.CountryOfOrigin = ”Venezuela”;
        Coffee1.Strength = 3;
        // Skapa Binding object som refererar till instans
        Binding coffeeBinding = new Binding();
        coffeeBinding.Source = coffee1;
        coffeeBinding´.Path = new PropertyPath(”Name”);
        // Lägg till binding till Text property på TextBlock
        Textblock1.SetBinding(TextBlock.TextProperty, coffeeBinding);

}
```

.easec

Knyta controls till samling

- I många scenarios vill du knyta till information i en samling av objekt.
- WPF inkluderar controls som är designade för att rendera samlingar.
- ListBox, ListView, ComboBox, TreeView, dessa är exempel på controls för samlingar.
- Ärver från klassen ItemsControl.

.easec

Knyta controls till samling (forts.)

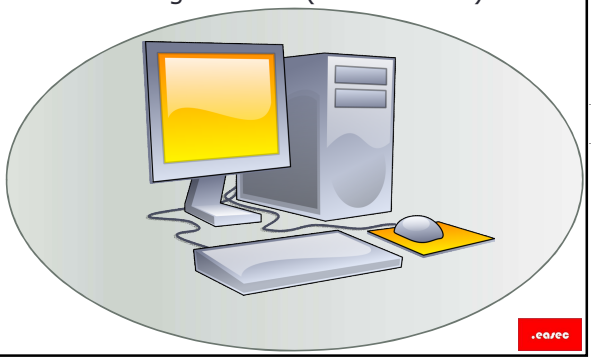
- För att knyta samling till klassen ItemsControl, behöver du:
  - Specificera källa för samling i egenskapen ItemsSource för ItemsControl.
  - Specificera egenskap i källa som skall visas i egenskapen DisplayMemberPath.
- Du kan knyta ItemsControl till vilken samling som helst, om samling implementerar interface IEnumerable.

.easec

Data Templates

- Används för att specificera hur dina komponenter renderas.
- Ger en möjlighet att kontrollera hur dina controlers renderas och visas.

Övning Skapa webbtjänst och användaregränssnitt (11b och 11c)



Lektion 3: Styla ett användaregränssnitt

- Skapa återanvändbara resurser i XAML.
- Resource Dictionary file.
- Definera styles som resurs.
- Använda Property Triggers.
- Skapa dynamisk omvandling.

Skapa återanvändbara resurser i XAML

- XAML ger möjlighet till att skapa resurser som är återanvändbara.
- Ger ett antal fördelar:
  - Resurs definieras på ett ställe och används på flera ställe.
  - Resurs redigeras på ett ställe, istället för på alla ställen där resurs används.
  - XAML-fil blir mindre och lättare att läsa.
- Varje WPF-kontroll har en Resource egenskap, till denna kan resurs läggas till.
- Kan också definieras i filen App.xaml.

.eorec

Skapa återanvändbara resurser i XAML -exempel

- Skapa resurs:

```
<Window x:Class="DataBinding.MainWindow"
        xmlns=http://schemas.microsoft.com/winfx/2006/xaml/presentation
        xmlns:x=http://schemas.microsoft.com/winfx/2006/xaml
        xmlns:loc="clr-namespace:DataBinding"
        Title="Återanvändbara resurser" Height="350" Width="525">
  <Window.Resources>
    <SolidColorBrush x:Key="MinPensel" Color="Coral" />
  ...
  </Window.Resources>
  ...
</Window>
```

```
<StackPanel>
    <SolidColorBrush x:Key="MinPensel" Color="Coral" />
  ...
  </Window.Resources>
  ...
</StackPanel>
```

.eorec

Skapa återanvändbara resurser i XAML -exempel

- Referera till resurs:

```
<StackPanel>
   <Button Content="Klicka här!" Background="{StaticResource MinPensel}" />
   <TextBlock Text="Foreground" Foreground="{StaticResource MinPensel}" />
   <TextBlock Text="Background" Background="{StaticResource MinPensel}" />
   <Ellipse Height="50" Fill="{StaticResouce MinPensel}" />
</StackPanel>
```

.eorec

Resource Dictionary file

- Om du behöver skapa ett antal återanvändbara resurser, kan det vara lättare att skapa en separat fil, resource dictionary file.
- Är en XAML-fil.
- Kan skapas via Add New Item, i Visual Studio.

Definera styles som resurs

- I många fall vill du applicera samma värde för egenskaper på control av samma typ.
- Exempelvis:
  - Sida som består av fem stycken textrutor och dessa skall se ut på samma sätt.
- Ett style element skapas som resurs i XAML.
- Ger dig möjlighet att applicera samling av värden till några eller alla controls av specifik typ.

Definera styles som resurs (forts.)

- Att defininera style:
  - Lägg till Style element till samling av resurser i din applikation (ex: Window.Resouces).
  - Använd attributet TargetType för Style elementet, för att specificera typ av control (ex: TextBox el Button).
  - Använd attributet x:Key för Style elementet för att ge möjlighet till control att specificera denna style. Alternativt hoppa över x:Key och då kommer style att appliceras på alla controls av samma typ.
  - I Style element, använd elementet Setter för att specificera specifika egenskaper.

Definera styles som resurs -exempel

- Skapa style:

```
<Window.Resources>
   <Style TargetType="TextBlock" x:Key="BlockStyle1">
     <Setter Property="FontSize" Value="20" />
     <Setter Property="Background" Value="Black" />
     <Setter Property="Foreground">
        <Setter.Value>
          <LinearGradientBrush StartPoint="0.5,0" EndPoint="0.5,1"
               <LinearGradientBrush.GradientStops>
                  <GradientStop Offset="0.0" Color="Orange" />
                  <GradientStop Offset="1.0" Color="Red" />
               </LinearGradientBrush.GradientStops>
          </LinearGradientBrush>
        </Setter.Value>
     </Setter>
   </style>
   ...
</Windows.Resources>
```

.eorec

Definera styles som resurs –exempel (forts.)

- Applicera style:

```
<TextBlock Text="Drick mer kaffe!" Style="{StaticResource BlockStyle1}" />
```

.eorec

Använda Property Triggers

- Använd triggers för att applicera egenskaper för style baserat på villkor:
  - Använd elementet Trigger för att identifiera ett villkor.
  - Använd elementet Setter för att applicera förändring.

```
<Window.Resources>
   <Style TargetType="Button">
      <Style.Triggers>
        <Trigger Property="IsMouseOver" Value="True">
             <Setter Property="FontWeight" Value="Bold"> />
        </Trigger>
      </Style.Triggers>
   </Style>
   ...
</Window.Resources>
```

.eorec

Skapa dynamisk omvandling

- Använd EventTrigger för att identifiera händelse som skall starta animering.
- Använd Storyboard för att identifiera egenskaper som skall förändras.
- Använd DoubleAnimation för att definiera förändringar.

```
<Window.Resources>
   <Style TargetType="Button">
      <Style.Triggers>
         <Trigger Property="IsMouseOver" Value="True">
              <Setter Property="FontWeight" Value="Bold" />
         </Trigger>
      </Style.Triggers>
   </Style>
   ...
</Window.Resources>
```

.easee

Skapa dynamisk omvandling - exempel

```
<EventTrigger RoutedEvent="Image.MouseDown">
   <BeginStoryboard>
       <Storyboard>
          <DoubleAnimation
            Storyboard.TargetProperty="Height"
            From="200" To="300" Duration="0:0:2" />
        </Storyboard>
   </BeginStoryboard>
   ...
</EventTrigger>
```

.easee

Repetitionsfrågor

.easee