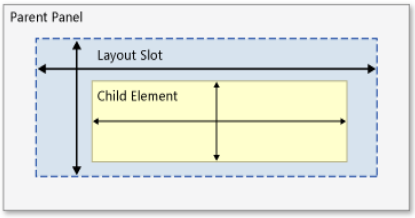
# Windows Phone 7 alapú szoftverfejlesztés ZH kérdések összefoglaló

## Silverlight

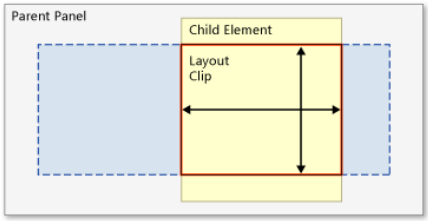
1. **Röviden ismertesse a XAML nyelvet!**
   A XAML egy XML-alapú deklaratív osztály példányosító nyelv a .NET Framework-höz, amelyet a WPF, Silverlight és WF technológiákkal szoktak használni. A .NET objektumokat és tulajdonságaikat (property) felelteti meg az XML tageknek és attribútumoknak, az XML névterek pedig egy vagy több .NET-es névtérnek. Mivel XML-alapú, ezért hierarchikus és minden elemnek egyetlen közvetlen szülője lehet.

2. **Mutassa be a Silverlight Layout életciklusának fázisait!**
   Measure (méretezési fázis): a szülő objektum (konténer elem) átadja minden gyermek elemének a számára kijelölt terület méretét. A vezérlő ezután kiszámolja, hogy mekkora területen kíván elhelyezkedni.
   Arrange (elrendezési fázis): a rendszer minden egyes gyermekelemnek átad egy **Rect** (x,y, szélesség, magasság) típusú objektumot, kijelölve ezzel mindegyiknek a pozícióját.

3. **Röviden ismertesse az alábbi fogalmakat: Bounding Box, Layout-slot, Layout-clip!**
   Bounding Box: a vezérlőt határoló téglalap.

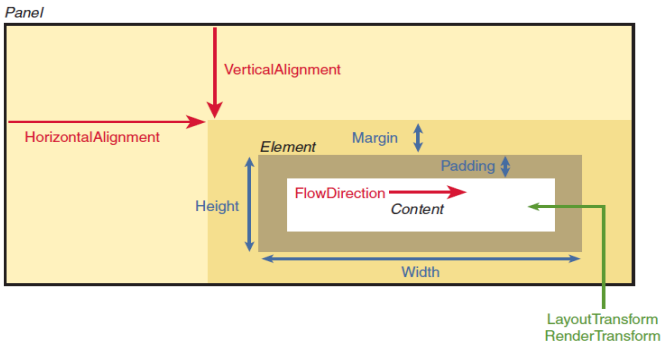



   Layout-slot: az a keret amit a vezérlő elfoglalhat. Kilóghat belőle ha a tartalma nagyobb.




   Layout-clip: a layout slotból kilógó elem látható része.

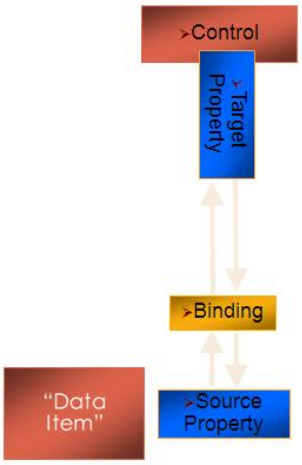
4. **Röviden ismertesse az alábbi fogalmakat: margin, padding (Ábra)!**
   Margin: az elem széleinek távolsága a szülő vagy egyéb vezérlők megfelelő széleitől, a Bounding Boxon kívül hagy extra helyet.
   Padding: a vezérlőn belüli elemek távolsága a panel széleitől.




5. **Röviden ismertesse a Dispatcher object működésének lényegét!**
   Silverlight alkalmazásokban a UI szál (UI thread) felelős a vezérlők megjelenítéséért, a hosszabb számításigényes feladatokat másik szál(ak)on szokás végezni (worker thread, Background Worker). Ezekről a szálakról viszont nem érhetők el közvetlenül a UI komponensek, ezért van szükség a Dispatcher osztályra. Az osztály BeginInvoke függvényének egy delegate-et átadva elérhetjük, hogy az adott kód a UI szálon fusson le, így elérhesse a UI komponenseket. A CheckAccess metódussal ellenőrízhetjük, hogy az aktuális kód futasákor a UI szálon vagyunk-e. Csak a DispatcherObject osztály leszármazottjain alkalmazható (ez őse minden UI osztálynak, ez biztosít egy Dispatcher nevű property-t).

6. **Röviden ismertesse a legfontosabb transzformációkat (forgatás, skálázás, torzítás, mozgatás)!**
   A transzformációk a UI objektumok RenderTransform tulajdonságán keresztül adhatók meg:
   RotateTransform: adott fokkal való elforgatás (Angle, CenterX, CenterY).
   Scaletransform: skálázás, vagyis adott pontból való kicsinyítés/nagyítás (ScaleX, ScaleY).
   SkewTransform: döntés/torzítás (SkewX, SkewY).
   TranslateTransform: víszintes/függőleges eltolás (X, Y).

7. **Mutassa be az adatkötés lényegét, és folyamatát Silverlight platformon (Ábra is kell)!**
   Adatkötés: egy vezérlő tulajdonságainak (dependency preopery-k) hozzákötése az adatforráshoz (annak tetszőleges property-jéhez).




   Adatkötés lépései:
   - adatforrás definiálása

- felület tervezése
- üzleti objektum kivezetése a felületre

**8. Röviden ismertesse az adatkötés módjait Silverlight platformon!**

Egyszeri (One-time): egyszer megy végbe szinkronizáció, amikor a felület betöltődik.Ezt követően nem tükrözi a forrás változásait és a forrás sem változik meg ha a vezérlőnek a hozzákötött tulajdonságát megváltoztatjuk.


Egyirányú (One-way): ha módosul az adatforrás, akkor a vezérlő is megjeleníti a változást, de visszafelé nem működik. Ez az alapértelmezett kötési mód.


Kétirányú (Two-way): olyan mint az egyirányú, de itt már vezérlő tulajdonságának megváltozása is maga után vonja a célobjektum változását.


**9. Mutassa be az INotifyPropertyChanged interéfszt, valamint a ObservableCollection-t!**

INotifyPropertyChanged: akkor kell implementálni ha olyan forrás property-hez akarunk kötni, amely nem Dependency Property. Ha az adott tulajdonság megváltozik, akkor meg kell hívni a PropertyChanged eseményt, ezzel jelezve a Binding Engine felé, hogy frissítenie kell a céltulajdonság értékét.

```
public interface INotifyPropertyChanged
{
    event PropertyChangedEventHandler PropertyChanged;
}
```

ObservableCollection: generikus lista, amely implementálja az INotifyPropertyChanged és az INotifyCollectionChanged interfészeket (tehát tartalmazza a PropertyChanged és CollectionChanged eseményeket), így jelez a Binding Engine felé ha a listához új elemet adnak, törölnek belőle, vagy frissülnek az elemei.

**10. Röviden ismertesse a Binding objektum legfontosabb tulajdonságait!**

A Binding osztály az adatkötés jellemzőit írja le. Főbb tulajdonságai:

- Source: az adatforrás objektum neve
- RelativeSource: ha az adatforrást a célforráshoz relatíve akarjuk megadni (saját tulajdonsághoz akarjuk kötni)
- ElementName: ha az adatforrás egy másik vezérlő
- Mode: az adatkötés módja (One-time, One-way, Two-way)
- Path: az adatforrás azon property-jének neve amelyhez hozzákötjük a cél property-t

- Converter: az IValueConverter interfészt megvalósító osztály egy példánya, amely az adatcsere előtti transzformációért felelős (Convert és ConvertBack metódusok)

**11. Röviden ismertesse az adatkötés során az adatok validációjával kapcsolatos lehetőségeket!**

Validáció: (kétirányú) adatkötéskor vizsgálhatjuk meg vele az adatok helyességét, jelezhetünk vele hiba esetén. Beállításai:

| Property | Leírás | Default érték |
|---|---|---|
| ValidatesOnExceptions | ha a forrás property beállítása során kivétel keletkezik, akkor a Binding Engine létrehoz egy ValidationError-t és hozzáadja a hibák listájához, ami megjelenik a grafikus felületen | false |
| NotifyOnValidationError | meghívja-e a BindingValidationError eseményt kivétel keletkezése esetén | false |
| ValidatesOnDataError | ha a forrás property beállítása az IDataError implementációja hibát jelez, akkor a Binding Engine létrehoz egy ValidationError-t és hozzáadja a hibák listájához, ami megjelenik a grafikus felületen | false |
| ValidatesOnNotifyDataErrors | meghívja-e a BindingValidationError eseményt INotifyDataError által kiváltott hiba esetén | true |

Validation.Errors: ez a gyűjtemény tartalmazza az adatkötés során előfordult hibákat. Akik módosíthatják:

- a cél/forrás property setter metódusa
- az adatkötéshez rendelt Converter
- IDataErrorInfo.Item
- INotifyDataErrorInfo.GetErrors( )

Lehetőségek a validációra:

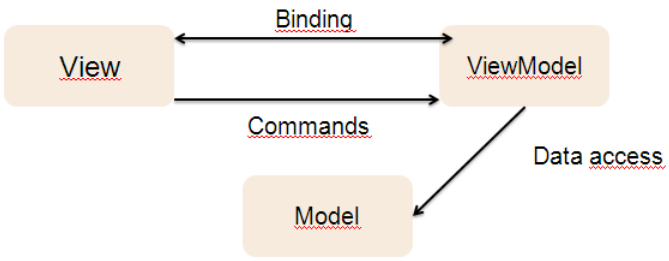
- **feliratkozás a BindingValidationError eseményre** (ha ValidatesOnExceptions = true és NotifyOnValidationError = true), amely a Routed Event-ek közé tartozik, így bármelyik szülő elemben elkapható (bubble)
- **IDataErrorInfo**: custom validáció valósítható meg vele egy helyen, ha implementáljuk az üzleti objektumban (ValidatesOnDataError = true kell hogy legyen, ha azt akarjuk a grafikus felületen is megjelenjen a hibajelzés)
- **INotifyDataError**: komplex custom validációt tesz lehetővé, támogatja az aszinkron validációt is, ha implementáljuk az üzleti objektumban (ValidatesOnNotifyDataErrors = true kell hogy legyen, ha azt akarjuk a grafikus felületen is megjelenjen a hibajelzés)

**12. Röviden ismertesse az erőforrás fogalmát!**

Olyan tetszőleges típusú objektumok, amelyeket nem csak lokálisan, hanem egymástól függetlenül több helyen is fel kívánunk használni. Pl: Brush, Font, stb.

Lehet statikus vagy dinamikus (utóbbi csak WPF-ben támogatott). Az objektumok Resources tulajdonságában definiálhatók vagy külön ResourceDictionary-ben.
Több szinten is létrehozhatjuk: vezérlő, Page, App (globális).
Az x:Key attribútummal megadott kulccsal hivatkozhatunk rájuk.

**13. Röviden ismertesse az MVVM mintát (Ábra is kell)!**




- Model:
  - entitás osztályok, adatforrások, DTO-k
  - Adatok betöltése, mentése, felhasználói felülettől független komponensek
- View:
  - Felhasználói felület leírása
  - Csak UI specifikus kódot tartalmazhat
- ViewModel:
  - View aktuális állapotának a leírása
  - Állapotkezelés
  - View-n keletkezett események (VM oldalon funkcióhívás) kezelése
  - View-n keletkezett inputok kezelése
  - Modell-ből a szükséges adatok előállítása

**14. Milyen problémákra nyújt megoldás az MVVM minta?**

Külön választja a megjelenítést a logikától, így segíti a Designer-Developer együttműködést: a designerek a View-n, a developerek a ViewModelen tudnak párhuzamosan dolgozni. A szétválasztás emellett könnyebben átláthatóvá és karbantarthatóvá teszi a projektet, könnyebb a tesztelés is.

**15. Hogyan biztosítja a ViewModellek függetlenségét az MVVM minta?**

A ViewModellek a Messanger singleton objektumon keresztül kommunikálnak egymással lazán csatolt módon, üzeneteken keresztül (Register, UnRegister és Send metódusok). Pl:

```
// Üzenetküldés
 public void RemoveBook()
    {
      dataSource.RemoveBook(BookID);
      Messenger.Default.Send(new BookRemovedMessage(BookID));
    }
```

```
// Regisztrálás
Messenger.Default.Register<BookRemovedMessage>(this, message =>
        {
          foreach (var item in Books)
          {
            if(item.BookID==message.BookID)
            {
               Books.Remove(item);     break;
            }
 }
});
```

16. **Hogyan biztosítja a View és a Viewmodell laza csatolását az MVVM minta?**
    A Commanding minta alkalmazásával, így a Viewnak nem kell ismernie a ViewModellt, ahhoz hogy a metódusait hívhassa: az adott parancsot a vezérlő Command tulajdonságához kötjük.

```
private RelayCommand loadBooksCommand;
public RelayCommand LoadBooksCommand
{
        get
        {
          if (loadBooksCommand == null)
            loadBooksCommand = new RelayCommand(LoadBooks);
          return loadBooksCommand;
        }
}
```

```
<Button Content="Load Books"
          Command="{Binding LoadBooksCommand, Mode=OneWay}"></Button>
```

17. **Röviden ismertesse a stílusok fogalmát!**
    **Stílus: újrahasznosítható erőforrás, egy adott vezérlőtípusra vonatkozó tulajdonság értékek (Setter) halmaza. A FrameworkElement osztály minden leszármazottja rendelkezik egy Style nevű property-vel, ezen keresztül állítható be egy vezérlő stílusa.**

```
<Style TargetType=”Button” x:Key=”MyButtonStyle”>
          <Setter Property=”FontFamily” Value=”Arial”/>
          <Setter Property=”Background” Value=”Green”/>
</Style>
```

18. **Röviden ismertesse a Silverlightban támogatott animáció típusokat!**

| Property type | Corresponding basic (From/To/By) animation | Corresponding key-frame animation |
|---|---|---|
| **Color** | **ColorAnimation** | **ColorAnimationUsingKeyFrames** |
| **Double** | **DoubleAnimation** | **DoubleAnimationUsingKeyFrames** |
| **Point** | **PointAnimation** | **PointAnimationUsingKeyFrames** |
| **Object** | **None** | **ObjectAnimationUsingKeyFrames** |

19. **Röviden ismertesse az animációk működését Silverlightban!**

A vezérlők tulajdonságait (Dependency Property-ket) lehet animálni Silverlightban, C#/VB.NET kódból és XAML-ből létrehozhatók. Az animációk külön szálon futnak (render thread).

Két típusa van:

- From / To / By: egyszerű kezdő- és végérték közötti animáció
- Key-frame alapú: kulcspontokat adhatunk, azok között definiálhatunk tranzíciókat (Discrete, Linear, Spline)

Animációhoz tartozó fontosabb osztályok:

- Timeline: minden animáció ebből származik, fontosabb tulajdonságai: Duration, SpeedRatio, RepeatBehavior, AutoReverse.
- StoryBoard: konténer az animációs objektumok számára, fontosabb tulajdonságai: TargetName, TargetProperty. Fontosabb metódusai: Begin, Pause, Resume, Stop.

```
<Canvas.Resources>
        <Storyboard x:Name="myStoryboard">
        <!-- Animate the center point of the ellipse. -->
        <PointAnimation Storyboard.TargetProperty="Center"
                Storyboard.TargetName="MyAnimatedEllipseGeometry"
                Duration="0:0:5" From="20,200" To="400,100"
                RepeatBehavior="Forever" />
        </Storyboard>
</Canvas.Resources>
<Path Fill="Blue">
        <Path.Data> <!-- Describes an ellipse. -->
                <EllipseGeometry x:Name="MyAnimatedEllipseGeometry" Center="20,20"
                RadiusX="15" RadiusY="15" />
        </Path.Data>
</Path>
```

## Orientáció, navigáció

**20. Röviden mutassa be a Panoramic, Pivot vezérlőket!**

**Panoramic**: adott témakörrel kapcsolatos gyűjtőoldal, amelynek csak egy szeletét látja egyszerre a felhasználó (jobbra-balra scrollozható). Megnöveli a képernyő méretét így több információt tudunk elhelyezni.



```
<controls:Panorama Title="Outback Adventure">
        <controls:Panorama.Background>
             <ImageBrush Stretch="Fill" ImageSource="/panorama.jpg"/>
        </controls:Panorama.Background>
            <controls:PanoramaItem Header="Recent">
                <StackPanel CacheMode="BitmapCache">
                    <TextBlock TextWrapping="Wrap" Text="Kakadu"
                               Style="{StaticResource PanoramaTextBlock}" />
                     <TextBlock TextWrapping="Wrap" Text="Cairns"
                                Style="{StaticResource PanoramaTextBlock}" />
                     <TextBlock TextWrapping="Wrap" Text="Alice Springs"
                                Style="{StaticResource PanoramaTextBlock}" />
                     <TextBlock TextWrapping="Wrap" Text="Darwin"
                                Style="{StaticResource PanoramaTextBlock}" />
                </StackPanel>
            </controls:PanoramaItem>
            <controls:PanoramaItem Header="Photos">
                <ListBox ItemsSource="{Binding Collection}"
                         Style="{StaticResource PanoramaImageListBox}"
                         Height="500"/>
            </controls:PanoramaItem>
</controls:Panorama>
```

**Pivot**: azonos adatnak a különböző nézeteit szokás megjeleníteni vele (ez is jobbra balra scrollozható), a Tab controlhoz hasonló.



```
<controls:Pivot Title="MY APPLICATION">
    <!--Pivot item one-->
    <controls:PivotItem Header="item1">
        <Grid/>
    </controls:PivotItem>
    <!--Pivot item two-->
    <controls:PivotItem Header="item2">
        <Grid/>
    </controls:PivotItem>
</controls:Pivot>
```

**21. Röviden ismertesse az mobil orientáció detekciójának lehetőségeit!**

A PhoneApplicationPage osztály Orientation tulajdonságával kérdezhető le.

```
public enum PageOrientation
{
        None = 0,
        Portrait = 1,
        Landscape = 2,
        PortraitUp = 5,
        PortraitDown = 9,
        LandscapeLeft = 18,
        LandscapeRight = 34
}
```

Fel lehet iratkozni a PhoneApplicationPage osztály OrientationChanged eseményére:

```
private void Page_OrientationChanged(object sender, OrientationChangedEventArgs e)
 {
        if (this.IsInLandscape())
        {
          MessageBox.Show("Landscape");
        }
        else
        {
          MessageBox.Show("Portrait");
        }
}
```

**22. Röviden ismertesse az application bar-t!**

A képernyő alján elhelyezkedő, legfeljebb 4 gombot (IconButton) tartalmazó. A menüelemekkel ellentétben az IconButton-ok kis helyet foglalnak el, grafikusak (48x48-as méretűek) és a leggyakrabban használt funkciók elérésére szokás használni.


```
<phone:PhoneApplicationPage.ApplicationBar>
 <shell:ApplicationBar IsMenuEnabled="False" StateChanged="ApplicationBar_StateChanged">
   <shell:ApplicationBar.MenuItems>
     <shell:ApplicationBarMenuItem Text="Clear" x:Name="ClearMenuItem"
                                   Click="ClearMenuItem_Click"/>
       <shell:ApplicationBarMenuItem Text="Opacity 0" x:Name="Opacity0"
                                     Click="Opacity0_Click"/>
       <shell:ApplicationBarMenuItem Text="Opacity 0.5" x:Name="OpacityHalf"
                                     Click="OpacityHalf_Click"/>
       <shell:ApplicationBarMenuItem Text="Opacity 1" x:Name="Opacity1"
                                     Click="Opacity1_Click"/>
   </shell:ApplicationBar.MenuItems>
   <shell:ApplicationBarIconButton Text="convert" IconUri="/icons/Percent.png" IsEnabled="True"
                                   Click="ConvertToPercent_Click"/>
 </shell:ApplicationBar>
</phone:PhoneApplicationPage.ApplicationBar>
```

**23. Röviden ismertesse a navigációval kapcsolatos alapfogalmakat (NavigationService, Fragment, QueryString)**

**NavigationService**: ez az osztály felelős a megadott oldal megkereséséért és betöltéséért, valamint az ehhez kapcsolódó események elsütéséért. Navigate függvényével tudunk tetszőleges oldalra elnavigálni:

```
this.NavigationService.Navigate(new Uri("SecondPage.xaml",UriKind.Relative));
```

A Navigating eseményére feliratkozva megakadályozhatjuk a navigációt, ha a NavigatingCancelEventArgs argumentum Cancel tualjdonságát true-ra állítjuk.

**Fragment**: navigálás során a # karakter után megadhatunk extra adatot adhatunk meg (maximum egyet).

```
private void NavigateWithFragmentButton_Click(object sender, RoutedEventArgs e)
```

```
{
      this.NavigationService.Navigate(
               new Uri("SecondPage.xaml#HelloWorld", UriKind.Relative));
}

protected override void OnFragmentNavigation(FragmentNavigationEventArgs e)
{
            base.OnFragmentNavigation(e);
            MessageBox.Show("Fragment: " + e.Fragment);
}
```

**QueryString:** egy vagy több paraméter átadása a címben.

```
private void NavigateWithQueryButton_Click(object sender, RoutedEventArgs e)
{
    this.NavigationService.Navigate(
           new Uri("/SecondPage.xaml?CustomerId=1234&Product=555",UriKind.Relative));
}

protected override void OnNavigatedTo(NavigationEventArgs e)
{
     base.OnNavigatedTo(e);
     foreach (var item in NavigationContext.QueryString)
     {
        MessageBox.Show("Query String [" + item.Key + "] = " + item.Value);
     }
}
```

**24. Milyen lehetőségek vannak a navigáció során adatok átadására két page között?**

- Fragment-en keresztül
- QueryString használata
- Globális változó / App szintű tároló

**25. Hogyan működik a vissza gomb? Hogyan tudjuk befolyásolni a működését?**

Hardveres gomb tartozik a funkcióhoz. Alapértelmezett működés:

- ha az alkalmazás első oldalán nyomtuk le, akkor bezárul az alkalmazás és az előző program nyílik meg
- ha nem az első oldalon vagyunk, akkor visszaugrik az előzőre
- ha egy Dialog/Context Menu ki van nyitva, akkor bezárja azt, de nem navigál utána vissza

Az OnBackKeyPress virtuális függvény felülírásával reagálhatunk az eseményre.
Kódból is meghívható, a CanGoBack tulajdonsággal lekérdezhetjük, hogy visszaléphetünk-e:

```
private void BackIfWeCanButton_Click(object sender, RoutedEventArgs e)
{
            if (this.NavigationService.CanGoBack)
            {
                this.NavigationService.GoBack();
            }
}
```

**26. Ismertesse navigációhoz kapcsolódó legfontosabb page eseményeket (4 db)?**

- OnNavigatingFrom: közvetlenül azelőtt hívódik meg hogy elhagynánk az oldalt
- OnNavigatedFrom: akkor hívódik meg amikor az adott oldalról elnavigált a rendszer

- OnNavigatedTo: akkor hívódik meg miután az adott oldalra navigált a rendszer
- OnFragmentNavigation: ha átnavigált az oldalra és kapott egy fragment paramétert, az új oldalon kapjuk az eseményt

## Launceherek, chooserek, gesztúrák

### 27. Ismertesse a Task fogalmát!

A telefonon lévő számos funkció gyűjtőneve, melyeket két csoportba sorolnak:

- Launcher: olyan funkció, ami egy telefonon lévő másik alkalmazást indít el, semmilyen információt nem ad vissza a hívó alkalmazásnak
- Chooser: olyan funkció, ami vissza ad valamilyen információt a hívó alkalmazásnak.

### 28. Ismertesse a Taskok hívásának általános sémáját!

1) Task példány létrehozása
2) Szükséges property-k beállítása
3) Show() hívása

```
EmailAddressChooserTask addressTask = new EmailAddressChooserTask();
this.addressTask.Completed += addressTask_Completed;
addressTask.Show();

void addressTask_Completed(object sender, EmailResult e)
{
      if (e.TaskResult == TaskResult.OK)
      {
          EmailAddressText.Text = e.Email;
      }
}
```

### 29. Soroljon fel és röviden mutasson be 5 db launchert!

| LAUNCHERS | DESCRIPTION |
|---|---|
| EmailComposeTask | Composes a new email. |
| PhoneCallTask | Initiates a phone call to a specified number. |
| SmsComposeTask | Composes a new text message. |
| SearchTask | Launches Bing Search with a specified search term. |
| WebBrowserTask | Launches Internet Explorer browsing to a specific URL. |
| MarketplaceDetailTask | Launches Marketplace with the details of a specific application. |
| MarketplaceHubTask | Launches Marketplace at one of the three hubs: Applications, Music or Podcasts. |
| MarketplaceReviewTask | Launches Marketplace to provide a review of the current application. |
| MarketplaceSearchTask | Launches Marketplace and performs a search for content. |
| MediaPlayerLauncher | Launches Media Player. |

### 30. Soroljon fel és röviden mutasson be 5 db choosert!

| CHOOSERS | DESCRIPTION | RETURN TYPE |
|---|---|---|
| CameraCaptureTask | Opens camera application to take a photo. | PhotoResult |
| PhotoChooserTask | Selects an image from your Picture Gallery. | PhotoResult |
| EmailAddressChooserTask | Selects an e-mail address from your Contacts List. | EmailResult |
| PhoneNumberChooserTask | Selects a phone number from your Contacts List. | PhoneNumberResult |
| SaveEmailAddressTask | Saves an e-mail address to an existing or new contact. | |
| SavePhoneNumberTask | Saves a phone number to an existing or new contact. | |

**31. Röviden ismertesse a gesztúrákhoz kapcsolódó alapfogalmakat (tap, double-tap, pan, flick, touch-and-hold, pinch and strech)! Mindegyik alkalmazására mondjon egy-egy példát!**

| Esemény | Leírás | Példa |
|---|---|---|
| tap | a Click eseménynek felel meg (lenyomás + felengedés) | gombra való kattintás |
| double-tap | Két Tap esemény gyorsan egymás után | Zoom-in/Zoom-out funkciók megvalósítása |
| pan | Húzzuk az ujjunkat különböző irányokban | drag & drop, scrollolás |
| flick | Gyorsan mozgatjuk az ujjunkat valamilyen irányba. Vagy akkor kezdődik amikor a felhasználó először megérinti a képernyőt, vagy egy pan eseményt követhet. | miután elhagyja a felhasználó ujja a felületet még adott ideig mozognak a vezérlők lassuló ütemben |
| touch-and-hold | Kiválaszt egy vezérlőt és ott tartja a kezét a felhasználó | Context Menu, Tooltip |
| pinch and strech | Több ujjal érintjük a kijelzőt és mozgatjuk, maximum 4 konkurens pontot tud kezelni | Nagyítás funkció |

# Bing map, médiakezelés

**32. Röviden mutassa be a GeoCordinate osztályt (legfontosabb tulajdonságok)!**

Egy földrajzi koordinátát ír le. Fontosabb tulajdonságai:

- Latitude: szélesség
- Longitude: hosszúság
- Altitude: magasság
- Course: északtól való eltérés fokban
- HorizontalAccuracy: vízszintes pontosság méterben

- VerticalAccuracy: függőleges pontosság méterben
- IsUnknown: tartalmaz-e koordinátát?
- Speed: a pont sebessége méter/second

**33. Röviden mutassa be a GeoCordinateWatcher osztály működését!**

Egy csomagoló osztály a Windows Phone Location Service szolgáltatás köré. Hozzáférhetünk vele a mobiltelefon nyers GPS és WIFI adataihoz, valamint a cellainformációkhoz. Beállítható a lekérdezés pontossága és gyakorisága.
Fontosabb metódusok:

- Start: elindítja az adatok figyelését a Location Service-ben
- Stop: leállítja az adatok figyelését a Location Serviceben

Tulajdonságok:

- DesiredAccuracy: Adatok pontossága
- MovementThreshold: a minimális elmozdulás az egyes PositionChanged események között
- Permission: engedélyezve van-e a hozzáférés?
- Position: GeoPosition objektum
- Status: a szolgáltatás státusza

Események:

- PositionChanged: pozíció változása
- StatusChanged: státusz változása

```
GeoCoordinateWatcher geowatcher;

private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
{
    geowatcher = new GeoCoordinateWatcher();
    geowatcher.StatusChanged += geowatcher_StatusChanged;
    geowatcher.PositionChanged += geowatcher_PositionChanged;
}

private void geowatcher_PositionChanged(object sender,
                        GeoPositionChangedEventArgs<GeoCoordinate> e)
{
    var locationText = PositionString(e.Position.Location);
    this.Dispatcher.BeginInvoke(
                  () => this.GeoLocationText.Text = locationText);
}
```

**34. Röviden ismertesse a MapView-t!**

Elérhetők vele a Bing Maps szolgáltatásai (statikus képek, geocoding, routing, stb.), használatához be kell szerezni egy kulcsot (Application Key) a Bing Maps Portal-ról.
Fontosabb tulajdonságai:

- Center: Location koordináta, a térkép közepe
- Zoom level: nagyítás mértéke, területenként eltérő lehet
- Heading: 0 – 360 fok, mi az északi irány
- Pitch: -90, 90 közötti érték, milyen szögből nézzük a térképet

```
<Grid.Resources>
    <Microsoft_Phone_Controls_Maps:ApplicationIdCredentialsProvider
      ApplicationId="Av2HkDep0HWsgM2jzqwfbvfoAsimaoZOF4xMGghNzTW2cHfiS4vQ7fh
      GfLZ48Xnw" x:Key="MapCredentials" />
 </Grid.Resources>
```

```
<Microsoft_Phone_Controls_Maps:Map x:Name="BingMap" ZoomLevel=„15"
  Center="{Binding MapCenter, Mode=TwoWay}" ScaleVisibility="Collapsed„
  CredentialsProvider="{StaticResource MapCredentials}"/>
```

**35. Milyen lehetőségek vannak media lejátszására Phone 7 platformon?**

MediaElement: Silverlight komponens hang/videó lejátszására, egyszerre csak egy lehet belőle aktív.
SoundEffect/SoundEffectInstance: XNA komponens, hang lejátszásra.
MediaPlayer: XNA komponens, hang lejátszására.

**36. Röviden ismertesse a MediaElement osztályt! (Példa is)**

Silverlight komponens hang/videó lejátszására, egyszerre csak egy lehet belőle aktív. Képes médiát lejátszani a készüléken lévő fájlból, de akár az internetről is.
Fontosabb tulajdonságok:

- AutoPlay: boolean, betöltődés után azonnal induljon-e a lejátszás
- CurrentState: Closed, Opening, Buffering, Playing, Paused, Stopped
- Position: aktuális TimeSpan érték, lehet tekerni, ha a CanSeek = true
- Source: media URI-ja
- Volume: 0-1 (default: 0.5)

Fontosabb metódusok:

- Pause
- Play
- SetSource
- Stop

```
<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
  <MediaElement x:Name="AudioPlayer" HorizontalAlignment="Left"
                VerticalAlignment="Top" Source="test.mp3„ AutoPlay="True"
                Visibility="Collapsed" />
</Grid>
```

**37. Röviden ismertesse a SoundEffect osztályt! (Példa is)**

**SoundEffect: XNA komponens, az audio adatokat és meta adatokat tartalmazza, betölthetjük vele a kívánt audio fájlt.**
**SoundEffectInstance: le tudjuk játszani a SoundEffect-ben tárolt audió fájlt, az instance-ek osztozhatnak egy SoundEffect példányon.**

```
// Betöltés:
var sound = SoundEffect.FromStream(TitleContainer.OpenStream("blockrock.wav"));
// Instance létrehozása:
SoundEffectInstance soundInstance;
soundInstance = soundEffect.CreateInstance();
// Lejátszás:
soundInstance.Play();
```

**38. Röviden ismertesse a MediaLibrary osztályt!**

**Hozzáférhetünk vele a telefonon lévő media tartalmakhoz (kivéve a DRM-el védettekhez).**
**Fontosabb tulajdonságok:**

- **Albums**
- **Artists**
- **Generes**
- **Pictures**
- **Playlist**
- **Songs**
- **SavedPictures**

```
var library = new Microsoft.Xna.Framework.Media.MediaLibrary();
this.MediaList.ItemsSource = library.Songs;

private void MediaList_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
            if (e.AddedItems.Count == 0) return;
            var song = e.AddedItems[0] as Song;
            if (song == null) return;
            MediaPlayer.Play(song);
}
```

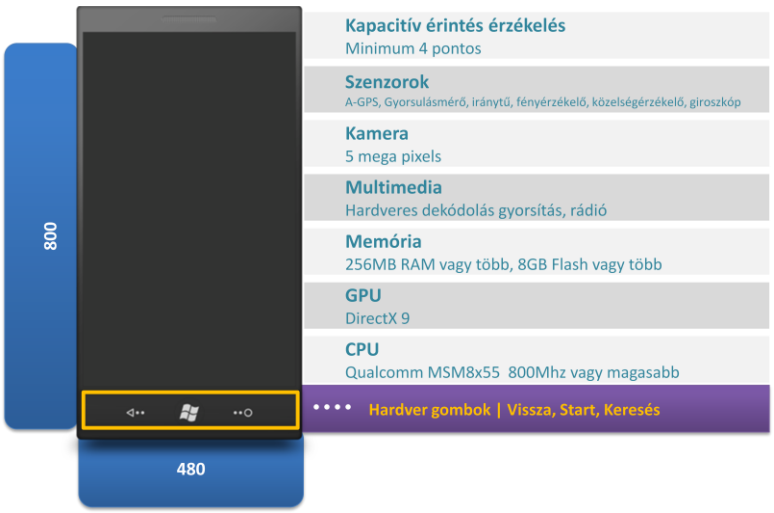
**39. Röviden ismertesse a FrameworkDispatcher osztály Update metódusát! Mikor kell használni?**

A metódus triggereli az események feldolgozását az XNA keretrendszeren belül, a hívás hatására a XNA keretrendszer üzenetsorában lévő üzenetek továbbításra kerülnek. Amennyiben Silverlighton belül használjuk, nekünk kell gondoskodni róla: periodikusan meg kell hívni (DispatcherTimer, 50 ms megfelelő intervallum). Érdemes rá külön osztályt írni és az App.xaml fájlban példányosítani.

# Architektúra

**40. Ismertesse a platform múltját, hardver jellemzőit, felépítését, és röviden a két fő fejlesztési keretrendszerét.**

A Windows CE beágyazott operációs rendszerből fejlődött ki, közvetlen elődje a Windows Mobile 5 (6.5).




Két fejlesztési keretrendszere:

- Silverlight: eredetileg egy böngésző plugin, a WPF „kis testvére”, jól használható grafikus felhasználói felületek leírására, deklaratív leíró nyelvvel rendelkezik (XAML)
- XNA: nagy teljesítményű grafikus alkalmazások és játékok fejlesztésére

**41. Írja le a METRO dizájn nyelv fő irányelveit, hasonlítsa össze más irányokkal, adjon javaslatokat használatára!**

A hangsúly a tipográfián és a tartalmon.
Egyértelműen felismerhető, csak a lényeget ábrázoljuk.
Minél kevesebb dizájn elem, ne vonja el a figyelmet.
Mint a metróban vagy reptéren.
Minimalizált szimbólumok – mindent elhagyni addig amíg még mindig érthető marad.

**42. Mutassa be a Pivot és Panorama vezérlőket a METRO szemszögéből!**

**Multitaszk**

43. Jellemezze a Windows Phone 7.5-re készült alkalmazások életciklusát, eseménykezelést, a fejlesztői teendőket!
44. Adjon áttekintést a háttér zene lejátszásának módszeréről!
45. Ismertesse a Background Agent-ek működését, típusait, fejlesztésének módját!
46. Ismertesse a letöltés kezelő (BTS) működését fejlesztői szempontból!
47. Mutassa be az időzített értesítések és emlékeztetők jellemzőit!
48. Mutassa be az élő csempéket, azok használatát!
49. Ismertesse a Push Notification működését!

**Adatkezelés**

50. Ismertesse az IsolatedStorage használatát és a .NET-es folyamkezelést!
51. Rajzolja le az Linq to SQL CE architektúrális felépítését és rövden adja meg az egyes komponensek szerepét!
52. Ismertesse a lekérdezések módját és a DataContext használatát a Linq to SQL CE közegben!
53. Vázolja a Linq to SQL CE ORM lehetőségeit, adjon példákat mindkettő használatára!
54. Mutassa be a Windows Phone 7.5 szenzorait, egyedi jellemzőiket és a programozási modellt!
55. Vázolja fel, hogy egy többrétegű telefonos alkalmazás milyen módon valósíthatja meg a felhasználó biztonságos autentikációját OAuth protokoll támogató közösségi hálózatokba Microsoft technológiai környezetben!