

# Bevezetés a Silverlight 3 programozásába

A felhasználói élmény fokozására törekvő webes technológiák között a következő lépés a Microsoft Silverlight. A Silverlight célja az, hogy a webes alkalmazások esetében ugyanazt a hitelességet és minőséget nyújtsa, mint ami az asztali alkalmazások felhasználói felületénél tapasztalható, így a webfejlesztők és -tervezők ügyfeleik különleges igényeihez szabva készíthetnek alkalmazásokat. Úgy tervezték, hogy a közös munkaformátum révén képes áthidalni a tervezők és a fejlesztők közti technológiai szakadékot. A böngésző XML-alapú formátumot képez le, így könnyű benne sablonokat készíteni, illetve automatikusan legenerálni. A formátum neve XAML (eXtensible Application Markup Language – kiterjeszhető alkalmazás jelölőnyelv).

A XAML előtt a webtervezők az eszközök egész tárházát hozták létre, amelyekkel hasonló technológiát használó arcuatcsaládot valósítottak meg. Ezután a fejlesztő, megkapva tőlük az anyagot, azt a saját választása szerint értelmezte. A tervezetet nem mindig sikerül megfelelően vagy problémamentesen átadni a fejlesztőknek, akinek néha számos módosítást kell végrehajtaniuk rajta, így sok esetben kompromisszumos megoldás születik. A Silverlighttal a tervező olyan eszközöket használhat, amelyek a tervet XAML-ben írják le, így a fejlesztő egyszerűen beépítheti a kódba, és telepítheti.

A Silverlight böngésző- és platformfüggetlen beépülő modul, amely sokoldalú médiaélményt és gazdag interaktív internetes alkalmazásokat biztosít a weben. Teljes programozási modellt kínál, amely támogatja az AJAX-ot, a .NET-keretrendszert és az olyan dinamikus nyelveket, mint a Python vagy a Ruby. A Silverlight 1.0 a jelenlegi webes technológiákkal, például AJAX-szal, JavaScripttel vagy Dynamic HTML-lel (DHTML) programozható. A Silverlight 2 már tartalmaz dinamikus és .NET-nyelvi támogatást, valamint temérdek olyan új funkcióval bővült, amely csak a .NET-keretrendszer használata estén érhető el – ilyen például az elszigetelt tároló, a hálózatkezelés, a gazdag vezérlőelem-készlet stb.

Könyvünk bemutatja, hogy a Silverlight 2 fent felsorolt összetevői mellett mi minden áll rendelkezésünkre, valamint megismertet a Silverlight legújabb verziójával, a Silverlight 3-mal.

A könyv első része bevezet a Silverlight alapjaiba, bemutatja az elérhető tervező- és fejlesztőeszközöket, második felében a programozási modellt vesszük jobban szemügyre.

## A Silverlight és a felhasználói élmény

A Silverlightot úgy tervezték, hogy egy olyan, jóval nagyobb ökoszisztéma része legyen, amely a lehető legjobb végfelhasználói élményt nyújtja. Számos olyan tipikus helyzet van, amikor információt próbálunk elérni az interneten:

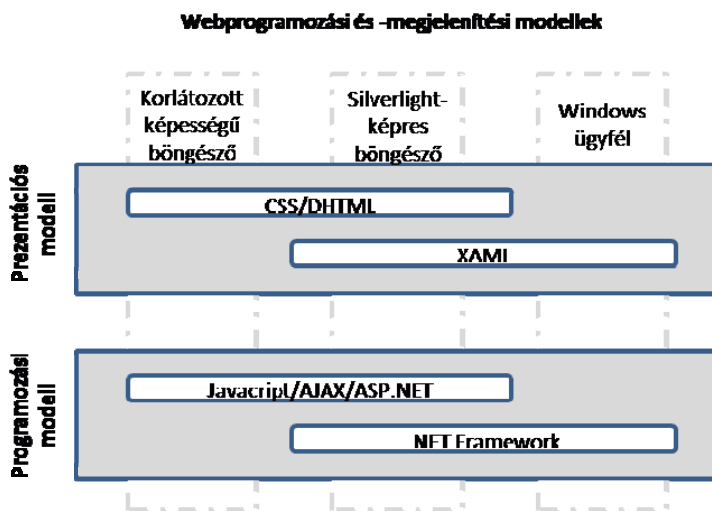
- mobil eszközök,
- otthoni szórakoztatóelektronikai termékek,
- csökkentett képességű böngészők (beépülőmodulok nélkül),
- kibővített böngészők (olyan beépülőmodulokkal, mint a Flash, a Java vagy a Silverlight),
- asztali alkalmazások,
- irodai programcsomagok.

Az évek folyamán változtak az ilyen alkalmazások működésével kapcsolatos felhasználói elvárások. Például az asztali számítógépen használt alkalmazásnak nagyobb felhasználói élményt kell nyújtania, mint egy ugyanolyan típusúnak egy mobil eszközön, ugyanis felhasználóként ahhoz szoktunk hozzá, hogy az asztali gépekben jóval nagyobb teljesítmény áll rendelkezésünkre. Továbbá számos felhasználó úgy gondolja, hogy mivel az alkalmazás az interneten található, nem rendelkezhet ugyanolyan kapacitással, mint egy hasonló asztali alkalmazás. Nekik például kisebb elvárásaik vannak egy web-alapú levelezőprogrammal szemben, mert nem hiszik, hogy ugyanazt a levelezési teljesítményt tudja nyújtani, mint az irodai programcsomagok által kínált megoldások (például a Microsoft Office Outlook).

Azonban, ahogy a platformok egyre inkább közelítenek egymáshoz, a felhasználói elvárások is nőnek – a *gazdag* jelzőt most már általánosan használják olyan élmény jellemzésére, amely a jelenlegi alap elvárási szint fölött van. A *gazdag internetes alkalmazás* kifejezést például arra a fokozott kifinomultsági szintre válaszul alkották meg, amellyel az internetezők az AJAX-alapú alkalmazásoknál találkoztak: ezek az e-mailezés vagy az online térképek területén dinamikusabb élményt nyújtanak. Az elvárások ilyen irányú fejlődése oda vezetett, hogy napjainkban a felhasználók olyan, még gazdagabb élményt követelnek, amely nemcsak a programmal szemben támasztott működésbeli és hatékonysági igényeknek felel meg, de a felhasználók egy vállalat termékeivel és szolgáltatásaival kapcsolatos észlelt elégedettségét is figyelembe veszi. Ez hosszán tartó kapcsolatot igényelhet a felhasználó és a cég között.

Ennek eredményeképpen a Microsoft elkötelezte magát a *felhasználói élmény* (UX – User eXperience) mellett, és olyan eszközöket, illetve technológiákat biztosít, amelyekkel a fejlesztők gazdag UX-alkalmazásokat készíthetnek. Továbbá ezeket az eszközöket konzisztensen tervezték, azaz a felhasználóiélmény-központú alkalmazások gyakorlata átvihető legyen az asztali- és a webalkalmazás-fejlesztés területei között. Így, ha gazdag asztali alkalmazást készítünk, de szükségünk van egy webes változatra is, a kettő átjárható. Ehhez hasonlóan, ha mobil alkalmazást készítünk, és szükségünk van egy internetes verzióra is, akkor nincs szükség kétféle gyakorlatra, kétféle eszközkészletre, illetve két fejlesztői gárdára.

Az 1.1. ábra mutatja a webre manapság elérhető prezentációs és programozási modelleket. Ahogy láthatjuk, a prezentációs modellben és a JavaScript, az AJAX és az ASP.NET fejlesztési modellben a böngészőalapú fejlesztési technológiák, a CSS és a DHTML a tipikusak. A .NET-keretrendszer 3.x verziójával rendelkező asztali gépek esetében a XAML biztosítja a prezentációs modellt, míg maga a keretrendszer a fejlesztési modellt. Ezek a modellek átfedik egymást, és a Silverlight-képes böngésző itt nyújtja a „mindkettőből a legjobbat” megközelítést.



1.1. ábra. Programozási és prezentációs modellek webre

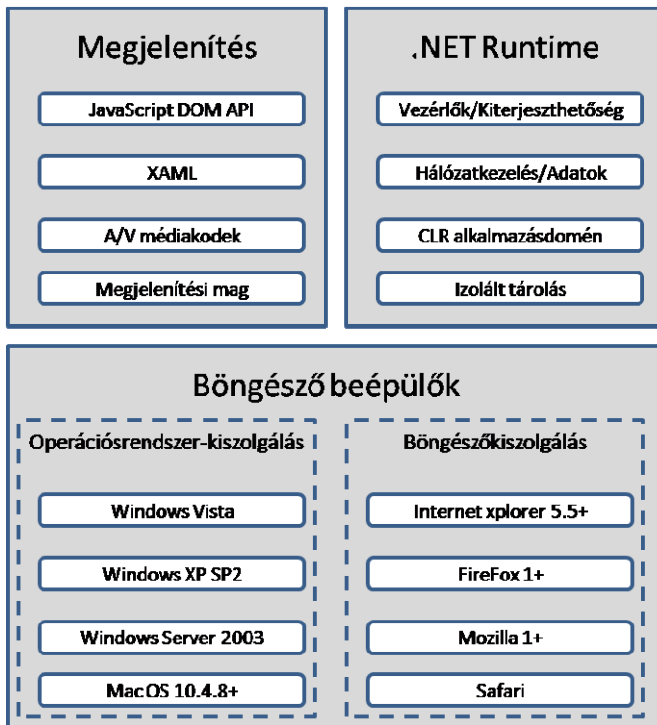
A tipikus gazdag interaktív alkalmazás olyan technológiákra épül, amelyek már léteznek a csökkentett képességű böngészők kategóriájában. A tipikus asztali alkalmazás a spektrum másik végén van, és ezekhez nem kapcsolódó technológiákat használ. A Silverlight-alapú böngésző azonban – amely biztosítja a CSS/DHTML és a XAML megjelenítési modellt, valamint a JavaScript/AJAX/.NET-keretrendszer programozási modellt – képes összekapcsolni ezeket a technológiákat egy olyan gazdag, de ugyanakkor vékonykliensalkalmazásban, amely a böngészőben fut.

A Silverlight egy böngészőbe beépülő modullal éri ezt el, amely olyan technológiákkal bővíti ki a böngésző tevékenységi körét, amelyek gazdag grafikus felületet biztosítanak (például idővonal-alapú animáció, vektorgrafika és audiovizuális média). Ezeknek a technológiáknak a használatát a Silverlight böngészőalapú XAML renderelési motorja teszi lehetővé. A gazdag grafikus felület megtervezhető XAML-ként, és mivel ez XML-alapú nyelv, és az XML csak szöveg, az alkalmazás kompatibilis a tűzfalakkal, valamint (potenciálisan) keresőmotor-barát. A böngésző megkapja a XAML-t, és megjeleníti azt.

Olyan technológiákkal kombinálva, mint az AJAX vagy a JavaScript, az alkalmazás kezelése lehet dinamikus folyamat – egyszerű JavaScript-programozással letölthetünk XAML-kódrészleteket, és hozzáadhatjuk azokat a felhasználói felületünkhöz, vagy szerkeszthetünk, átrendezhetünk olyan XAML-t, amely jelenleg megtalálható a megjelenítési fában, illetve el is távolíthatjuk őket.

# A Silverlight architektúrája

Ahogy az imént szó volt róla, a Silverlight alapműködését egy olyan böngészőbe beépülő modul biztosítja, amely megjeleníti a XAML-t, és vagy JavaScript- és böngészőalapú, vagy .NET-keretrendszer- és CLR- (Common Language Runtime – közös nyelvi futtatórendszer) alapú programozási modellt szolgáltat. Az ezt támogató architektúrát az 1.2. ábra mutatja.



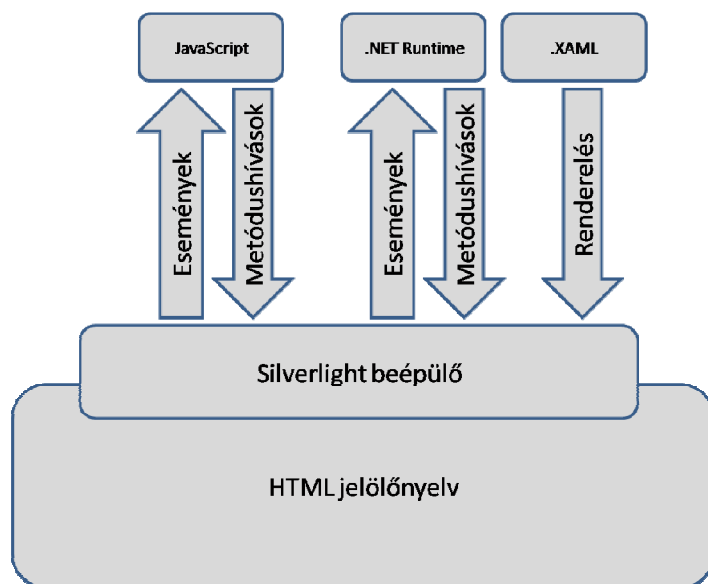
1.2. ábra. A Silverlight architektúrája

Amikor a böngésző vezérlőelemének szkriptjét írjuk, a Silverlight 1.0-ban látható fő programozási interfész a JavaScript DOM (dokumentum-objektummodell) API (alkalmazásprogramozási interfész) révén érhető el. A JavaScript DOM API használatával elkaphatunk az alkalmazásban felmerült felhasználói eseményeket (például egérmozgást vagy adott elemre kattintást), és válaszul futtathatunk egy kódot. Meghívhatunk metódusokat a JavaScript DOM-on, ezzel manipulálhatjuk a XAML-elemeket – például lehetővé téve egy médialejátszó vezérlését vagy animációk változtatását.

A gazdagabb és még hatékonyabb élmény érdekében programozhatunk olyan alkalmazást is, amelyet a vezérlőelem jelenít meg az új .NET CLR használatával. Ez a lehetőség – a JavaScript lehetőségein túl – felkínál számos, a .NET-keretrendszer részét alkotó névteret, így olyan dolgokat is megtehetünk, amelyeket JavaScriptben nagyon nehéz – ha nem lehetetlen – lenne, többek között hozzáférhetünk adatokhoz az ADO.NET, illetve a LINQ (Language-**I**ntegrated **Q**uery – kb. nyelvbe integrált lekérdezés) használatával, kommunikálhatunk webszolgáltatásokkal, készíthetünk és használhatunk egyedi vezérlőelemeket stb.

A prezentációs futtatókörnyezet ezen kívül rendelkezik a szükséges szoftverrel az olyan technológiákhoz, amelyek lehetővé teszik a H264, a Windows Media Video (WMV), a Windows Media Audio (WMA) és az MP3 formátumok lejátszását a böngészőben, mindenféle külső tartozék *nélkül*. Így például, a Macintosht használóknak WMV-tartalom lejátszásához nincs szükségük Windows Media Playerre – elég hozzá a Silverlight . A teljes prezentációs futtatókörnyezet alátámasztása a prezentációs kód, és ez kezeli az általános megjelenítési folyamatot. Ez mind része a böngésző-beépülőmodulnak, amelyet úgy terveztek, hogy támogassa a Windowson és a Macintoshon használható főbb böngészőket.

Az 1.3. ábrán egy olyan egyszerű alkalmazás architektúrája látható, amely Silverlight használatával fut a böngészőben.



1.3. ábra. Alkalmazásarchitektúra Silverlighttal

A böngészőben futó alkalmazások általában HTML-ből állnak. Ez a leírás vagy kódrészlet tartalmazza a Silverlight-beépülőmodul példányosításához szükséges hívásokat. A Silverlight-alkalmazás használata közben a felhasználók különböző eseményeket indítanak el, amelyeket akár JavaScript-, akár .NET-függvények alkalmazásával el lehet kapni. Másfelől, a programkód meghívhat metódusokat a Silverlight-tartalom elemeinek manipulálása, új tartalom hozzáadása, illetve létező tartalom eltávolítása érdekében. Végül a beépülőmodul olvashatja és renderelheti a XAML-t. Maga a XAML létezhet az oldalba illesztve, külső statikus fájlként vagy egy kiszolgálótól visszakapott dinamikus XAML-ként.

## A Silverlight és a XAML

A Silverlight architektúrájának és a tipikus alkalmazásfelépítésnek nagyvonalú bemutatása után vizsgáljuk meg a UX-et összetartó alapvető technológiát, a XAML-t.

A XAML egy XML-alapú nyelv, amely az alkalmazás vizuális készletét definiálja. Ide tartoznak a felhasználói felületek, a grafikai készletek, az animációk, a média, a vezérlőelemek stb. A Microsoft a Windows Presentation Foundation (WPF) – korábban Avalon – számára fejlesztette ki a XAML-t, amely kezdetben asztali gépekre szánt technológia volt, és a 3.0 verzió óta a .NET-keretrendszer része. Fejlesztésekor az volt a cél, hogy a tervezők és a fejlesztők közti szakadékot áthidalják.

A Silverlightban használt XAML abban különbözik a WPF-ben használttól, hogy ez főként a webes funkciókra koncentráló *részhalmoz*. Ezért, ha a WPF-ben mar megismerkedtünk a XAML-lel, észre fogjuk venni, hogy néhány címke és működés, például a `<Window>` elem, hiányzik.

A XAML XML-elemeket használ a grafikus felület definiálására. Minden Silverlight XAML dokumentum gyökere egy tároló elem, például egy *Canvas*, amely definiálja azt a területet, amelyen a grafikus felületet megrajzolják. Silverlight-webalkalmazás készítésekor rendelkezünk egy gyöker *Canvas* elemmel, amely a Silverlighthez szükséges XML-névtér-deklarációkat tartalmazza.

Íme, egy példa:

```
<Canvas
  xmlns="http://schemas.microsoft.com/client/2007"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  width="640" Height="480"
  Background="white"
>
</Canvas>
```

Figyeljük meg, hogy két névtér van deklarálva. A tipikus XAML-dokumentum elemek és attribútumok alap- és bővített halmazát tartalmazza, az utóbbi általában az *x:* prefixumot használja. A kiterjesztett névtértulajdonság egy példája a gyakran használt *x:Name*, amely nevet ad egy XAML-elemnek, így arra hivatkozhatunk a programkódban. A gyökérszintű *Canvas* elem deklarálja a névtér helyét ezek mindegyikéhez.

A *Canvas* elem egy tároló. Ez azt jelenti, hogy tartalmaz más, gyermek-elemeket. Ezek az elemek maguk is lehetnek más elemek tárolói, XML-dokumentumfaként határoznak meg egy felhasználói felületet. A következő példa egy egyszerű XAML-dokumentum, amely tartalmaz egy számos gyermekkel rendelkező *Canvas*-t, és ezen gyermekek közül néhány szintén *Canvas* tároló:

```
<Canvas
  xmlns="http://schemas.microsoft.com/client/2007"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  width="640" Height="480"
  Background="Black"
>
  <Rectangle Fill="#FFFFFF" Stroke="#FF000000"
    width="136" Height="80" Canvas.Left="120" Canvas.Top="240"/>
  <Canvas>
    <Rectangle Fill="#FFFFFF" Stroke="#FF000000" width="104" Height="96"
      Canvas.Left="400" Canvas.Top="320"/>
    <Canvas width="320" Height="104" Canvas.Left="96" Canvas.Top="64">
      <Rectangle Fill="#FFFFFF" Stroke="#FF000000" width="120"
        Height="96"/>
      <Rectangle Fill="#FFFFFF" Stroke="#FF000000" width="168"
        Height="96" Canvas.Left="152" Canvas.Top="8"/>
    </Canvas>
  </Canvas>
</Canvas>
```

Láthatjuk, hogy a gyökérszintű *Canvas* két gyermekkel rendelkezik: egy *Rectangle* és egy másik *Canvas* elemmel. Ez a második *Canvas* szintén tartalmaz egy *Rectangle* és egy *Canvas* elemet, míg az utolsó *Canvas* tartalma két *Rectangle*. Ez a hierarchikus struktúra lehetővé teszi a vezérlőelemek logikai csoportosítását, valamint a közös elrendezés és más viselkedések megosztását.

A Silverlight XAML számos olyan alakzatot támogat, amelyek komplex objektumokká kombinálhatók. A rendelkezésünkre álló egyszerű alakzatok a következők (a XAML használatának részletes leírása a 4. fejezetben – „A Silverlight XAML alapjai” – található):

- **Rectangle:** téglalap alakzatot definiál a képernyőn,
- **Ellipse:** ellipszist vagy kört definiál,
- **Line:** két pontot összekötő vonalat rajzol,
- **Polygon:** sokszöget rajzol,
- **Polyline:** sok szakaszt rajzol,
- **Path:** nemlineáris vonal (mint egy firkálás) rajzolását teszi lehetővé.

A XAML ezen felül támogat *ecseteket* is, amelyek azt definiálják, hogy az objektum hogy lesz kifestve a képernyőn. Egy objektum belső területét *kitöltő* ecsettel festjük ki, míg a körvonalát *vonallal* használatával rajzoljuk meg. Az ecseteknek több típusa van, köztük egyszínű, színátmenet, kép és video.

A következő példa egy ellipszis kitöltése *SolidColorBrush* használatával:

```
<Ellipse Canvas.Top="10" Canvas.Left="24" width="200" Height="150">
  <Ellipse.Fill>
    <SolidColorBrush Color="Black" />
  </Ellipse.Fill>
</Ellipse>
```

Ebben az esetben az ecset a Silverlight által támogatott 141 elnevezett szín egyikét, a *Black*et használja. Használhatjuk a szabványos hexadecimális RGB-színjelölést is egyedi színekhez.

A kitöltések és a vonalak fokozatos kitöltését egy színátmenetes ecset segítségével oldhatjuk meg. Az átmeneteket egy *normalizált területen* több *gradiensszakasz* alkalmazásával definiálhatjuk. Tehát, ha például lineáris gradiens szeretnénk balról jobbra – feketéből fehérbe átmenve különböző szürkeárnyalatokkal –, a szakaszokat egy normalizált vonal szerint határozhatjuk meg. Ebben az esetben a normalizált vonal kezdetét tekintjük a 0, a végét pedig az 1 pontnak. Így az egydimenziós térben lévő balról-jobbra fokozat megáll 0-nál, majd 1-nél. Ha olyan gradiens szeretnénk, amely kettőnél több színből áll – például fekete-piros-fehér –, akkor valahol 0 és 1 között definiálnunk kell egy harmadik állomást is.

Azt azonban ne feledjük, hogy kitöltéskor kétdimenziós térben dolgozunk, és ilyenkor a (0,0) jelzi a bal felső sarkot, míg (1,1) a jobb alsót. Ezért ahhoz, hogy fokozatos ecsettel töltsünk ki egy négyzetet, a következőképpen kell használnunk a *LinearGradientBrush*-t:

```
<Rectangle width="200" height="150" >
  <Rectangle.Fill>
    <LinearGradientBrush StartPoint="0,0" EndPoint="1,1">
      <LinearGradientBrush.GradientStops>
        <GradientStop Color="Red" Offset="0" />
        <GradientStop Color="Black" Offset="1" />
      </LinearGradientBrush.GradientStops>
    </LinearGradientBrush>
  </Rectangle.Fill>
</Rectangle>
```

A *TextBlock* elem révén a XAML a szövegek megjelenítését is támogatja. Az olyan tipikus szövegtulajdonságokat, mint a tartalom, betűtípus, betűméret, tördelés és egyebek, attribútumokon keresztül vezérelhetjük. Egy egyszerű példa:

```
<TextBlock TextWrapping="wrap" width="100">
  Hello there, how are you?
</TextBlock>
```

XAML-ben számos transzformációval manipulálhatjuk az objektumokat, a következők ezek közé tartoznak:

- **RotationTransform:** Adott szögben elforgatja az elemet.
- **ScaleTransform:** Objektumok nyújtására vagy kisebbítésére használható.
- **SkewTransform:** Adott irányban és adott mértékben elferdíti az objektumot.
- **TranslateTransform:** Adott vektor mentén mozgatja az objektumot.
- **MatrixTransform:** Olyan matematikai transzformáció létrehozására használható, amely kombinálni tudja az előzőeket.

A már meglévő transzformációk csoportosíthatók egy komplex transzformációba. Ez azt jelenti, hogy egyidejűleg mozgathatunk egy objektumot az áthelyezéssel, módosíthatjuk méretét az átméretezéssel, és el is forgathatjuk úgy, hogy az egyes transzformációkat csoportosítjuk. Íme, egy transzformációs példa, amely elforgatja és átméretezi a vásznat:

```

<Canvas.RenderTransform>
  <TransformGroup>
    <RotateTransform Angle="-45" CenterX="50" Centery="50"/>
    <ScaleTransform ScaleX="1.5" ScaleY="2" />
  </TransformGroup>
</Canvas.RenderTransform>

```

Az idővonal segítségével a XAML az animációkat is támogatja, meghatározva, hogy tulajdonságaik idővel hogyan változzanak. Ezek az idővonalak egy *forgatókönyvben* vannak. Az animációtípusok közé tartoznak a következők:

- **DoubleAnimation:** Lehetővé teszi olyan számtulajdonságok animálását, mint például azok, amelyek helymeghatározásra szolgálnak.
- **ColorAnimation:** Lehetővé teszi a színtulajdonságok, (például kitöltések) transzformálását.
- **PointAnimation:** Lehetővé teszi a kétdimenziós tereket definiáló pontok animálását.

A tulajdonságokat módosíthatjuk lineáris módon, ebben az esetben a tulajdonság értékei egy idővonalon vannak sorrendben meghatározva, vagy „kulskocka” módon, amikor több mérföldkő definiálása mentén történik az animáció. Mindezeknek a részletes leírása az 5. fejezetben („*Transzformáció és animáció a XAML-ban*”) található.

Az említetteken kívül teljes grafikus felületeket és elrendezéseket is definiálni fogunk XAML-vezérlőelemek segítségével, ezeknek a témaköröknek a részletes leírása a 9. fejezetben („*Összetett Silverlight-vezérlőelemek*”), valamint a 2. részben („*A Silverlight 3 programozása .NET-keretrendszerben*”) található.

## A Silverlight és az Expression Studio

A Microsoft Expression Studio egy tervezőknek és/vagy grafikusoknak szánt robusztus, modern eszközkészlet. Segítségével a tervezők ugyanazokkal az eszközökkel dolgozhatnak, mint a Microsoft Visual Studio eszközcsomagot alkalmazó fejlesztők.

Az Expression-csomagban többféle eszköz található:

- **Expression Web:** Webtervező eszköz, amelyben HTML-t, DHTML-t, CSS-t és más, szabványos webes technológiákat használhatunk webes alkalmazások tervezésére, készítésére és kezelésére.
- **Expression Media:** Ezzel a médiakészlet-kezelő eszközzel katalogizálhatjuk és rendszerezhetjük a készleteket, valamint kódolhatunk, illetve változtathatjuk a kódolást különböző formátumok között.
- **Expression Encoder:** Ezzel az alkalmazással médiakészletek kódolását kezelhetjük. Emellett arra is használható, hogy egybecsomagoljuk a médiát a releváns programkóddal, így lesz hozzá egy Silverlight-médi lejátszónk is.
- **Expression Design:** Illusztrációs és grafikai tervezőeszköz, amellyel grafikai elemeket, valamint készleteket készíthetünk webes és asztali alkalmazások grafikus felületéhez.
- **Expression Blend:** Ezzel az eszközzel XAML-alapú felhasználói felületeket és alkalmazásokat készíthetünk asztali gépekre a WPF, vagy webre a Silverlight használatával.

A Silverlight lehetővé teszi ezen alkalmazások közül egyesek, vagy akár az összes használatát. A fejezet a további részében bemutatjuk, hogy az Expression Design, az Expression Blend és az Expression Encoder hogyan bővíti eszközszerünket Silverlight-alkalmazások tervezéséhez és készítéséhez.

## A Silverlight és az Expression Design

Az Expression Design egy olyan grafikai tervezőeszköz, amellyel grafikai készleteket hozhatunk létre alkalmazásainkhoz. Mivel az Expression Design egy nagy és kifinomult eszköz, ebben a részben csak röviden tekintjük át, hogyan használható a Silverlight XAML-ben.

Az Expression Design segítségével keverhetjük a vektor- és a raszteralapú (bittérkép) képeket a teljes megvalósítás érdekében. Az Expression Design számos grafikai fájlformátumot importálni tud, ezek a következők:

- Adobe Illustrator – PDF-kompatibilis (\*.ai)
- Adobe Photoshop (\*.psd)
- Graphical Interchange Format (.gif)

- Portable Network Graphics formátum (.png)
- Bittérképek (.bmp, .dib, .rle)
- JPEG-formátumok (.jpeg, .jpg, .jpe, .jfif, .exif)
- Windows Media fotók (.wdp, .hdp)
- Tagged Image File Format (.tiff, .tif)
- Ikonok (.ico)

A következő képtípusokba lehet exportálni:

- XAML Silverlight vászon
- XAML WPF erőforráskönyvtár
- XAML WPF vászon
- Hordozható dokumentumformátum (.pdf)
- Adobe Photoshop (.psd)
- Tagged Image File Format (.tif, .tiff)
- JPEG-formátumok (.jpeg, .jpg)
- Windows-bittérkép (.bmp)
- Portable Network Graphics formátum (.png)
- Graphical Interchange Format (.gif)
- Windows Media fotók (más néven HD-fotó; .wdp)

Mint látjuk, az Expression Design támogatja a grafikai eszközök XAML-fájlként való exportálását. A fejezet későbbi részében látni fogjuk, hogy az Expression Design hogyan használható egyszerű alkalmazás grafikai elemeinek megtervezésére. Ezt követően a tervezetet XAML-ként fogjuk exportálni, és így az Expression Blendben, illetve a Visual Studióban már könnyen felhasználhatjuk egy alkalmazás létrehozására.

Az 1.4. ábra az Expression Design Export XAML-párbeszédablakát mutatja. Számos formátum használható, amelyek közül az egyik a XAML Silverlight vászon (az ábrán ez van kijelölve). Ez a lehetőség a rajzunkat olyan XAML-elemek részhalmazával formázza, amelyet a Silverlight felhasználhat, így az eredményül kapott XAML-t importálhatjuk a Visual Studióba vagy az Expression Blendbe saját Silverlight-alkalmazásunk elkészítéséhez.



1.4. ábra. XAML exportálása az Expression Designből

A tartalom olyan XML-dokumentumként lesz exportálva, amely a rajzunk elemeit tartalmazó *Canvas* elemet tartalmazza. Íme, egy (csonka) példa:

```
<?xml version="1.0" encoding="utf-8"?>
<Canvas xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" x:Name="Document">
  <Canvas x:Name="Layer_1" width="640.219" height="480.202" Canvas.Left="0"
Canvas.Top="0">
    <Ellipse x:Name="Ellipse" width="135" height="161" Canvas.Left="0.546544"
Canvas.Top="20.3998" Stretch="Fill" StrokeLineJoin="Round"
Stroke="#FF000000" Fill="#FF000000"/>
    <Path x:Name="Path" width="135.103" height="66.444"
Canvas.Left="-0.555986" Canvas.Top="-0.389065"
Stretch="Fill" StrokeLineJoin="Round" Stroke="#FF000000"
Fill="#FF000000" Data="..."/>
    <Path x:Name="Path_0" width="19.4583" height="23.9019"
Canvas.Left="75.8927" Canvas.Top="76.1198" Stretch="Fill"
StrokeLineJoin="Round" Stroke="#FF000000"
Fill="#FF000000" Data="..."/>
    <Path x:Name="Path_1" width="11.0735" height="24.0564"
Canvas.Left="60.473" Canvas.Top="106.4" Stretch="Fill"
StrokeLineJoin="Round" Stroke="#FF000000"
Fill="#FF000000" Data="..."/>
  </Canvas>
</Canvas>
```

```

<Path x:Name="Path_2" width="76" height="29.8274" Canvas.Left="31.5465"
      Canvas.Top="127.4" Stretch="Fill" StrokeThickness="7"
      StrokeLineJoin="Round" Stroke="#FF000000" Data="..."/>
<Path x:Name="Path_3" width="20.3803" height="27.1204"
      Canvas.Left="31.2028" Canvas.Top="75.306" Stretch="Fill"
      StrokeLineJoin="Round" Stroke="#FF000000"
      Fill="#FF000000" Data="..."/>
</Canvas>
</Canvas>

```

Ezt a XAML-t aztán bemásolhatjuk az Expression Blendbe vagy a Visual Studióba, és a grafikai elemet felhasználhatjuk alkalmazásunkban.

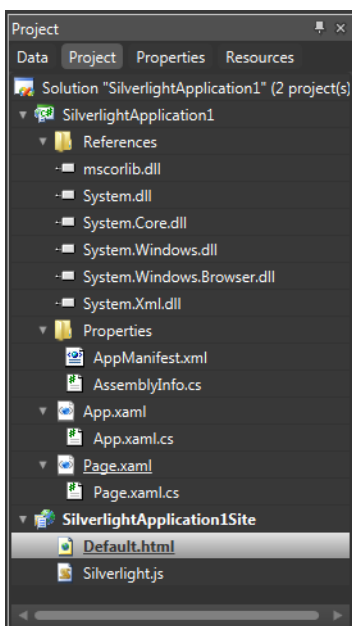
## A Silverlight és az Expression Blend

Az Expression Blend natív támogatással rendelkezik Silverlight-alkalmazások készítéséhez. Amikor elindítjuk az Expression Blendet és létrehozunk egy új projektet, két választási lehetőségünk van Silverlight-projektek készítéséhez:

- **Silverlight Application:** Ez a lehetőség olyan Silverlight-alkalmazás-sablont hoz létre, amely tartalmaz mindent ahhoz, hogy elkezdhesünk készíteni egy Silverlight-alkalmazást. A sablon tartalmazza a szükséges .NET-szerelvényeket, egy *properties* mappát az alkalmazás-manifesztummal, az App.xaml-t az alkalmazás belépési pontját definiáló mögöttes kóddal, valamint egy üres vásznat tartalmazó alapoldalt, a mögöttes kóddal együtt.
- **Silverlight Website:** Ez ugyanaz, mint a Silverlight-alkalmazás-sablon, de van benne egy webes projekt is egy olyan HTML-oldallal, amely beágyazza a Silverlight-alkalmazást a szükséges JavaScript-fájlokkal együtt.

### A Silverlight Website projekt felfedezése

Amikor létrehozunk egy új Silverlight Website projektet az Expression Blendben, lesz benne egy alapértelmezett HTML-fájl, amely tartalmazza az összes szükséges JavaScriptet a Silverlight-vezérlőelem példányosításához, valamint a Silverlight.js fájl egy másolatát, amely a Silverlight szoftverfejlesztő készlet (Software Development Kit – SDK) része. Ez a fájl kezeli a Silverlight-beépülőmodul példányosítását és letöltését a felhasználók számára. A projektstruktúrát az 1.5. ábrán láthatjuk.



1.5. ábra. Projektstruktúra egy Silverlight Website projekthez

## Az alapértelmezett weblap

Az 1.1 lista az Expression Blend által a Silverlight-projekthez létrehozott egyszerű weblap kódjának egy részét mutatja.

1.1. lista: *Default.html* a Silverlight sablonból

```
<div id="silverlightControlHost">
<object data="data:application/x-silverlight,"
        type="application/x-silverlight-2" width="100%" height="100%">
  <param name="source" value="ClientBin/SilverlightApplication1.xap"/>
  <param name="onerror" value="onSilverlightError" />
  <param name="background" value="white" />
  <param name="minRuntimeVersion" value="2.0.31005.0" />
  <param name="autoUpgrade" value="true" />
  <a href="http://go.microsoft.com/fwlink/?LinkId=124807"
     style="text-decoration: none;">
    
  </a>
</object>
<iframe style='visibility:hidden;height:0;width:0;border:0px'>
</iframe>
</div>
</body>
```

A Silverlight-vezérlőelem példányosítása az `<object>` címkében történik. Ez az objektum több paramétert címkéz:

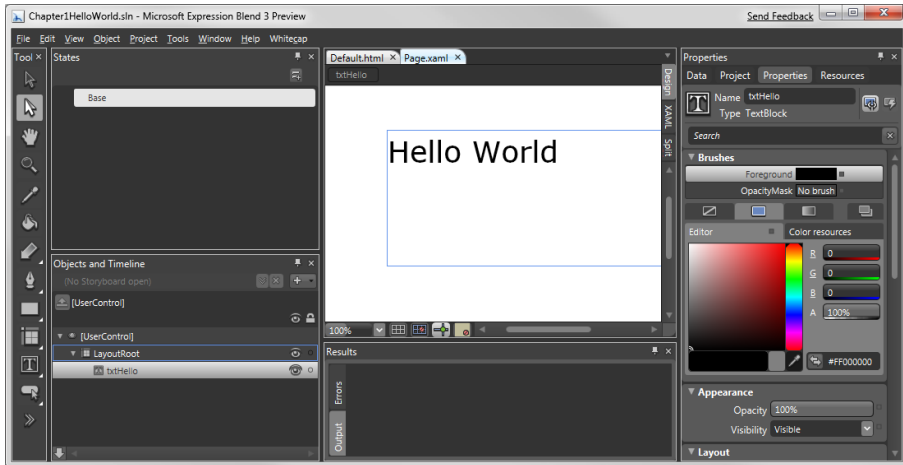
- Az első paraméter a forrás, amely a Silverlight-alkalmazáscsomag-fájltra (.xap) mutat, ez tartalmazza a lefordított Silverlight-alkalmazást. Ez lehet még hivatkozás egy statikus külső fájlra, egy olyan szolgáltatás URL-jére, amely képes XAML-t generálni, vagy egy nevesített szkript-blokkra a XAML-t tartalmazó oldalon.
- A második paraméter az *onerror*, amely egy JavaScript-blokkot definiál az oldalon, ez meghívásra kerül, ha a Silverlight-alkalmazás hibaüzenetet dob.
- A harmadik paraméter, a *background* definiálja a vezérlőelem háttérszínét, ha még nincs.
- A negyedik paraméter a *minRuntimeVersion*, amelyet a Silverlight az alkalmazás elindításához minimálisan szükséges verzióknak a vezérlésére használ. Tehát, ha például az alkalmazás nem használ Silverlight 3-specifikus funkciókat, akkor itt meg lehet adni a Silverlight 2-es verzióját (mint a listában), így a tartalom megtekintésekor a felhasználók nem lesznek rákényszerülve, hogy a Silverlight 3-ra frissítsenek.
- Ha az ötödik paraméter, az *autoUpgrade true* értékre van állítva, a Silverlight a példányosításkor automatikusan frissül. Ha *false* értékre van állítva, nem történik semmi.

## Silverlight-alkalmazás fejlesztése és futtatása az Expression Blendben

Az Expression Blend részletes leírása a 2. fejezetben („Silverlight-alkalmazás fejlesztése az Expression Blendben”) található. Ebben a részben gyors pillantást vehetünk arra, hogy az Expression Blend hogyan használható egyszerű Silverlight-alkalmazások készítésére és futtatására.

A New Project Wizard segítségével hozzunk létre egy Silverlight 3 weboldalt Chapter1HelloWorld néven. Nyissuk meg a Page.xaml-t a szerkesztőben, majd adjunk hozzá egy új szövegblokkot a tervezési felületen. Jelöljük ki ezt, és kattintsunk a Properties lapra.

A Properties lap beállításával adhatjuk meg, hogy nézzen ki a szöveg, például módosíthatjuk magát a szöveget és a betűméretet is (az 1.6. ábrán láthatunk egy példát).



1.6. ábra. Silverlight-projekt szerkesztése az Expression Blendben

Az Expression Blend 3-ban szerkeszthetjük a mögöttes kódot is – ez egy új funkció, amely az előző verziókban még nem volt.

Például, az 1.6. ábrán látható szövegblokknak eredetileg nem volt neve (a *Name* tulajdonság értéke <No Name> volt). Ezt módosíthatjuk *txtHello*-ra, majd nyissuk meg a *Page.xaml.cs* fájlt. Ehhez hasonló C#-kódot fogunk látni:

```
public Page(){
    // A változók inicializálásához elengedhetetlen.
    InitializeComponent();
}
```

Adjuk hozzá a következő sort az *InitializeComponent()* után:

```
txtHello.Text += " from Expression Blend";
```

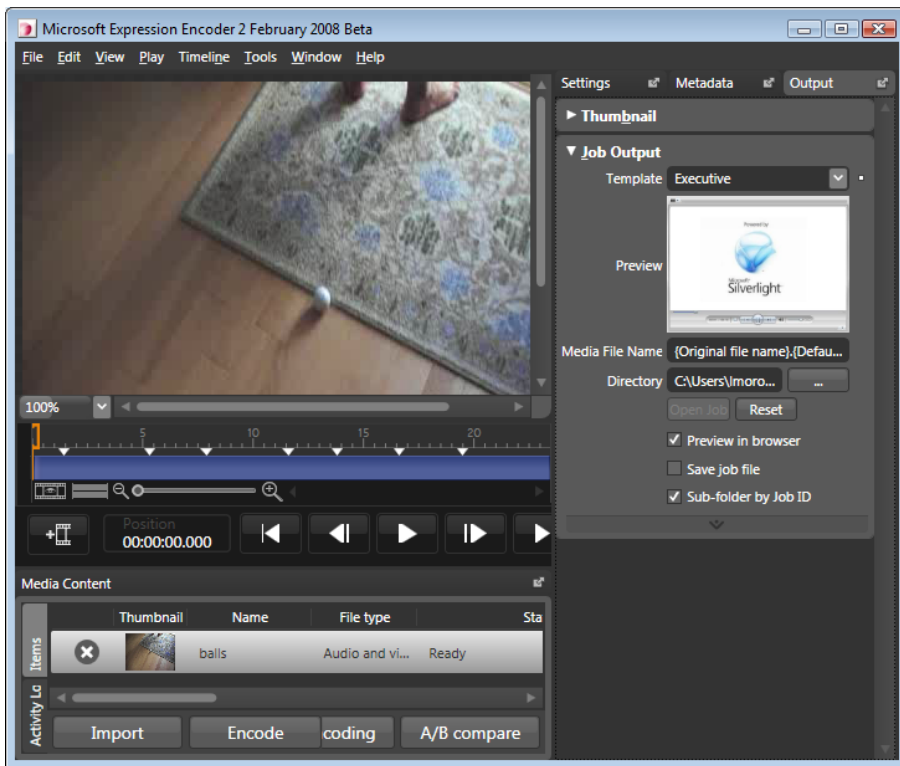
Ha most megnyomjuk az F5 billentyűt, látni fogjuk, hogy az Expression Blend lefordítja és becsomagolja a .NET-kódot, és megnyitja a böngészőt. A böngésző megjeleníti a Silverlight-tartalmat, amely a „Hello World from Expression Blend” szöveg.

A tervező „Hello World” szövegét a .NET-kód kiegészítette a „from Expression Blend” szöveggel.

Noha ez egy rendkívül egyszerű példa, remélhetőleg felcsigázza a kedves Olvasó érdeklődését a Silverlighttal megvalósítható számtalan különféle lehetőség iránt.

## A Silverlight és az Expression Encoder

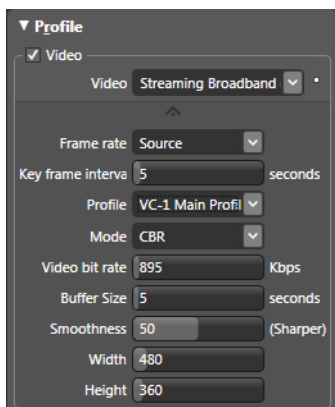
Az Expression Encoder segítségével a Silverlighttal videotartalmat kódolhatunk, bővíthetünk és tehetünk közzé. Az Expression Encoder felhasználói felülete összhangban van az Expression Studio csomag többi részével, illetve egy, köteget munkát lehetővé tevő parancssori interfésszel. Az Expression Encodert az 1.7. ábra mutatja.



1.7. ábra. Az Expression Encoder

Az Expression Encoderrel minden olyan formátumból importálhatunk videót, amelyhez létezik DirectShow szűrő, feltéve, ha az telepítve van számítógépünkre. Az Expression Encoder a lehívó ügyfélre optimalizált számos előre beállított profil egyikének használatával újrakódolja a videót VC-1-képes WMV-be. Az előre beállított profil tartalmazza a beállításokat az eszközök-höz, az interneten történő folyamatos adatátvitelhez, illetve igény szerinti tartalomszállításhoz.

Az előre beállított profilok nem korlátoznak – felülírhatunk bármilyen video-, illetve audiokódolási beállítást. Az 1.8 ábra mutat egy példát arra, hogyan állítható a videokódolás.



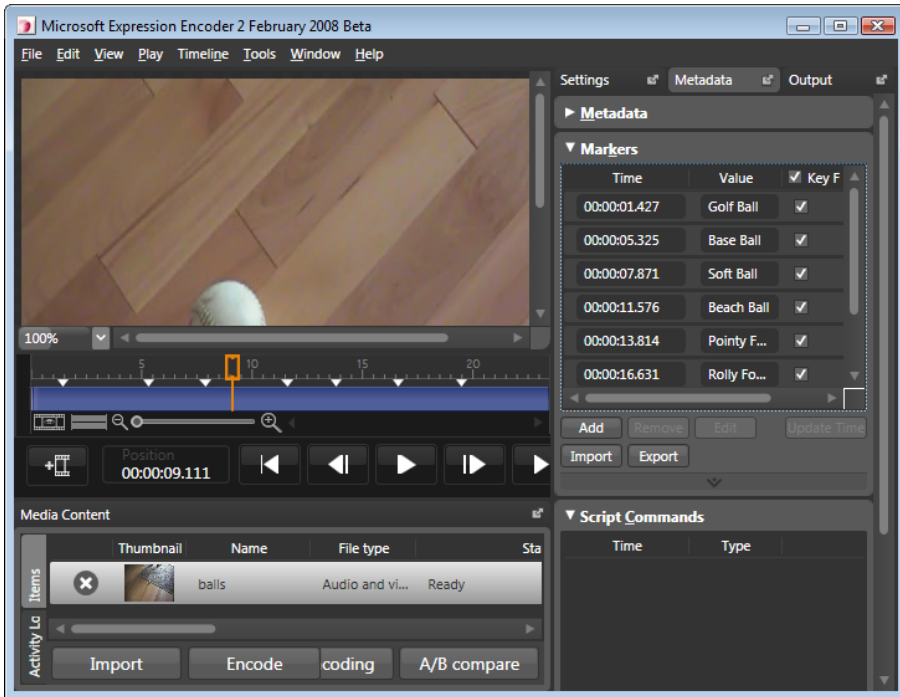
1.8. ábra. Videokódolási profil beállítása

Az Expression Encoder számos előre beállított médialejátszó alkalmazást tartalmaz a Silverlighthez. Ezek az alkalmazások „becsomagolják” a videót egy olyan Silverlight JavaScript-alapú alkalmazásba, amely bármely webkiszolgálón használható teljes, Silverlight-alapú megjelenítési élmény biztosítására.

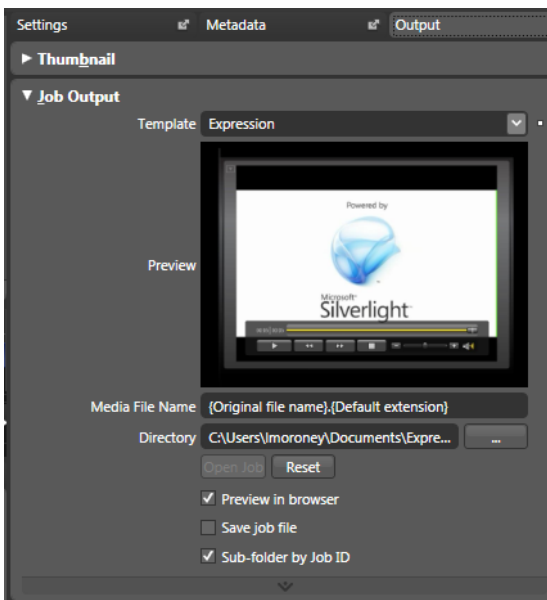
A kódolás mellett metaadatot is hozzáadhatunk a videóhoz. Klasszikus metaadatélmény, amikor a címkék bele vannak kódolva a videóba, az alkalmazás pedig reagál ezekre a címkékre. Az Expression Encoder segítségével nagyon egyszerű címkéket beszúrni. Egyszerűen húzzuk a kívánt pontra a lejátszófejet, válasszuk az Add Marker lehetőséget, majd írjuk be a megfelelő információt a jelölőhöz.

Ennek egy példáját az 1.9. ábrán, a képernyő jobb oldalán láthatjuk, ahol be van állítva a jelölő ideje, valamint a képernyőn abban a pillanatban látható labda típusa.

Az Output lapon választhatjuk ki a használni kívánt sablonlejátszót. Az 1.10. ábra mutatja az Expression-termékcsaládhoz illesztett sablont, amely az Expression-termékcsalád stílusjegyeit hordozza. Ha videolejátszót szeretnénk készíteni ezzel a sablonnal, egyszerűen importáljunk egy videót, válasszuk ki a sablont, majd kattintsunk az Encode gombra.

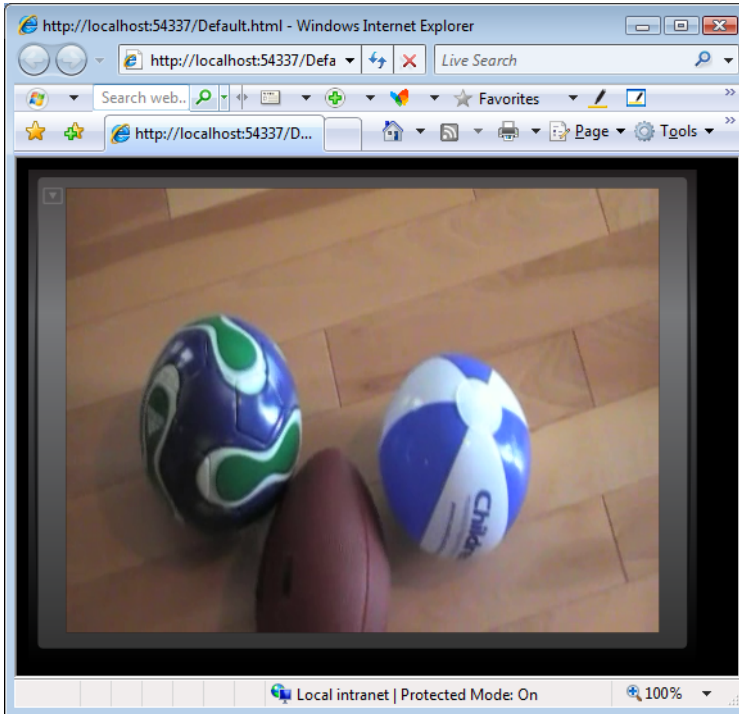


1.9. ábra. Jelölő hozzáadása egy adatfolyamhoz



1.10. ábra. Silverlight-médialejátzó készítése Expression Encoderrel

Miután kiválasztottuk a sablonlejátszót, teljes körű Silverlight-médialejátszót kapunk a videotartalmunkhoz. Az 1.11. ábrán láthatunk egy Silverlight-médialejátszót.



1.11. ábra. Az Expression Encoderrel készített médialejátszó

Ez a rész csak felszínesen érintette az Expression Encoder lehetőségeit, és azt, hogy ezek hogyan használhatók a Silverlighttel. További részletekért látogassa meg a [www.microsoft.com/expression](http://www.microsoft.com/expression) weboldalt.

## Összefoglalás

Ez a fejezet bemutatta a Silverlight 3-at, és általánosságban leírta, hogyan illik a web és a UX világába. Megtudtuk, hogy a Microsoft technológiája hogyan alkalmazható a jelenlegi UX-forgatókönyvekhez, vagy akár a „tipikus” UX-helyzetekre, és áttekintést kaptunk a Silverlight-architektúráról, a XAML-ről, valamint arról, hogyan lehet vele gazdag grafikus felületet létrehozni.

Ezekon kívül szó volt arról is, hogy a Microsoft Expression Suite hogyan egészíti ki Silverlight-alkalmazások készítésekor az olyan hagyományos fejlesztőeszközöket, mint a Visual Studio. Bemutattuk, hogyan használható az Expression Design grafikai készletek létrehozására, hogyan kapcsolhatók össze a készletek egyetlen interaktív alkalmazássá az Expression Blend segítségével, és hogyan használhatjuk az Expression Encodert videokészleteink kezelésére.

Ezek után itt az ideje jobban elmerülni témánkban: a következő néhány fejezetben a Silverlight API-ról lesz szó. A 2. fejezet az Expression Blend részletesebb vizsgálatával kezdődik, és leírja, a Silverlight hogyan használja azt.