Student Guide

Information in this document is subject to change without notice. Companies, names and data used in examples herein are fictitious unless otherwise noted. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Object Innovations.

Product and company names mentioned herein are the trademarks or registered trademarks of their respective owners.

Object Innovations is a registered trademark of Object Innovations.

Author: Robert J. Oberg

Special Thanks: Ernani Junior Cecon

Copyright ©2009 Object Innovations Enterprises, LLC  All rights reserved.

Object Innovations
877-558-7246
www.objectinnovations.com

Published in the United States of America.
# Table of Contents (Overview)

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 1</td>
<td>Introduction to Silverlight</td>
</tr>
<tr>
<td>Chapter 2</td>
<td>Programming Silverlight</td>
</tr>
<tr>
<td>Chapter 3</td>
<td>Using XAML</td>
</tr>
<tr>
<td>Chapter 4</td>
<td>Shapes and Lines</td>
</tr>
<tr>
<td>Chapter 5</td>
<td>Text and Fonts</td>
</tr>
<tr>
<td>Chapter 6</td>
<td>Brushes and Images</td>
</tr>
<tr>
<td>Chapter 7</td>
<td>Layout and Transforms</td>
</tr>
<tr>
<td>Chapter 8</td>
<td>Handling Events</td>
</tr>
<tr>
<td>Chapter 9</td>
<td>Animation and Media</td>
</tr>
<tr>
<td>Chapter 10</td>
<td>Networking</td>
</tr>
<tr>
<td>Chapter 11</td>
<td>Data Controls and Data Binding</td>
</tr>
<tr>
<td>Chapter 12</td>
<td>Data Access</td>
</tr>
<tr>
<td>Appendix A</td>
<td>Learning Resources</td>
</tr>
<tr>
<td>Appendix B</td>
<td>Silverlight and the Browser</td>
</tr>
</tbody>
</table>
Directory Structure

- Install the course software by running the self-extractor `Install_SilverCs_301.exe`.

- The course software installs to the root directory `C:\OIC\SilverCs`.
  
  - Example programs for each chapter are in named subdirectories of chapter directories `Chap01`, `Chap02` and so on.
  
  - The **Labs** directory contains one subdirectory for each lab, named after the lab number. Starter code is frequently supplied, and answers are provided in the chapter directories.
  
  - The **Demos** directory is provided for performing in-class demonstrations led by the instructor.

- **Data files install to the directory** `C:\OIC\Data`. 
# Table of Contents (Detailed)

## Chapter 1: Introduction to Silverlight
- What Is Silverlight? ................................................................. 3
- Silverlight and AJAX ............................................................... 4
- Silverlight and Flash ............................................................... 5
- Installing the Silverlight Plug-In ............................................ 6
- Silverlight Tools ........................................................................ 7
- Hello Silverlight ....................................................................... 8
- Allowing Blocked Contact ........................................................ 9
- Default.html ............................................................................ 10
- Page.xaml .................................................................................. 11
- XAML ....................................................................................... 12
- XHTML ..................................................................................... 13
- JavaScript Helper Functions .................................................. 14
- Using Silverlight.js ............................................................... 15
- createSilverlight.js .............................................................. 16
- Using createObjectEx() .......................................................... 17
- Width and Height in Per Cent ................................................... 18
- Providing for User Plug-In Download ..................................... 19
- Automatic Plug-In Download .................................................. 20
- Summary .................................................................................. 21

## Chapter 2: Programming Silverlight
- Silverlight Programming Models ............................................. 25
- Interactive JavaScript Applications ....................................... 26
- Using the Document Object Model ....................................... 27
- Default.html ............................................................................ 28
- JavaScript Code ...................................................................... 29
- XAML Files .............................................................................. 30
- Using Visual Studio 2008 ....................................................... 31
- Hosting the Silverlight Application ........................................ 32
- MainPage.xaml ....................................................................... 33
- MainPage.xaml.cs ................................................................. 34
- Designer Support ..................................................................... 35
- Layout Panels ......................................................................... 36
- SimpleHello Example ............................................................ 37
- Using a StackPanel ............................................................... 38
- Handling Events in Managed Code ....................................... 39
- Events with Visual Studio ...................................................... 40
- Code-Behind File ................................................................. 41
- Names of XAML Elements .................................................... 42
- Silverlight Controls ............................................................... 43
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spread Methods</td>
<td>167</td>
</tr>
<tr>
<td>Repeat and Reflect</td>
<td>168</td>
</tr>
<tr>
<td>Alpha Channel Changes</td>
<td>169</td>
</tr>
<tr>
<td>Painting Other Shapes</td>
<td>170</td>
</tr>
<tr>
<td>Radial Gradient Brushes</td>
<td>171</td>
</tr>
<tr>
<td>GradientOrigin</td>
<td>172</td>
</tr>
<tr>
<td>Image</td>
<td>173</td>
</tr>
<tr>
<td>Combining Transforms</td>
<td>229</td>
</tr>
<tr>
<td>Dynamic Skew Transform Example</td>
<td>228</td>
</tr>
<tr>
<td>Skew Transform</td>
<td>227</td>
</tr>
<tr>
<td>Dynamic Scale Transform Example</td>
<td>226</td>
</tr>
<tr>
<td>Scale Transform</td>
<td>225</td>
</tr>
<tr>
<td>Dynamic Rotate Transform Example</td>
<td>223</td>
</tr>
<tr>
<td>Rotate Transform</td>
<td>222</td>
</tr>
<tr>
<td>A Dynamic Transform</td>
<td>220</td>
</tr>
<tr>
<td>Transforms</td>
<td>218</td>
</tr>
<tr>
<td>Translate Transform</td>
<td>219</td>
</tr>
<tr>
<td>Controlling Size</td>
<td>196</td>
</tr>
<tr>
<td>Margin and Padding</td>
<td>197</td>
</tr>
<tr>
<td>Thickness Structure</td>
<td>198</td>
</tr>
<tr>
<td>Alignment</td>
<td>199</td>
</tr>
<tr>
<td>Size and Alignment Demo</td>
<td>200</td>
</tr>
<tr>
<td>Panels</td>
<td>207</td>
</tr>
<tr>
<td>Grid</td>
<td>208</td>
</tr>
<tr>
<td>Grid Example</td>
<td>209</td>
</tr>
<tr>
<td>Grid Demo</td>
<td>210</td>
</tr>
<tr>
<td>Star Sizing</td>
<td>211</td>
</tr>
<tr>
<td>Grid.ColumnSpan</td>
<td>212</td>
</tr>
<tr>
<td>Lab 7A</td>
<td>213</td>
</tr>
<tr>
<td>Positioning on a Canvas</td>
<td>214</td>
</tr>
<tr>
<td>Moving Elements</td>
<td>215</td>
</tr>
<tr>
<td>Z-Order</td>
<td>216</td>
</tr>
<tr>
<td>Clipping</td>
<td>217</td>
</tr>
<tr>
<td>Clipping</td>
<td>217</td>
</tr>
<tr>
<td>Spread Methods</td>
<td>167</td>
</tr>
<tr>
<td>Repeat and Reflect</td>
<td>168</td>
</tr>
<tr>
<td>Alpha Channel Changes</td>
<td>169</td>
</tr>
<tr>
<td>Painting Other Shapes</td>
<td>170</td>
</tr>
<tr>
<td>Radial Gradient Brushes</td>
<td>171</td>
</tr>
<tr>
<td>GradientOrigin</td>
<td>172</td>
</tr>
<tr>
<td>Image</td>
<td>173</td>
</tr>
<tr>
<td>Combining Transforms</td>
<td>229</td>
</tr>
<tr>
<td>Dynamic Skew Transform Example</td>
<td>228</td>
</tr>
<tr>
<td>Skew Transform</td>
<td>227</td>
</tr>
<tr>
<td>Dynamic Skew Transform Example</td>
<td>228</td>
</tr>
<tr>
<td>Combining Transforms</td>
<td>228</td>
</tr>
<tr>
<td>TransformGroup Example</td>
<td>230</td>
</tr>
<tr>
<td>Lab 7B</td>
<td>231</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Chapter 8: Handling Events</td>
<td>241</td>
</tr>
<tr>
<td>Events in Silverlight</td>
<td>243</td>
</tr>
<tr>
<td>Event Handlers in XAML</td>
<td>244</td>
</tr>
<tr>
<td>Simple Event Example</td>
<td>245</td>
</tr>
<tr>
<td>Adding/Removing Event Handlers</td>
<td>246</td>
</tr>
<tr>
<td>Dynamic Event Handling Example</td>
<td>247</td>
</tr>
<tr>
<td>Loaded Event</td>
<td>248</td>
</tr>
<tr>
<td>Mouse Events</td>
<td>249</td>
</tr>
<tr>
<td>Microsoft Silverlight Configuration</td>
<td>250</td>
</tr>
<tr>
<td>MouseEventArgs</td>
<td>251</td>
</tr>
<tr>
<td>Mouse Position Example</td>
<td>252</td>
</tr>
<tr>
<td>C# Code</td>
<td>253</td>
</tr>
<tr>
<td>Rubber-Band Drawing</td>
<td>254</td>
</tr>
<tr>
<td>Naïve Implementation</td>
<td>255</td>
</tr>
<tr>
<td>drawRect() Helper Method</td>
<td>258</td>
</tr>
<tr>
<td>Final Implementation</td>
<td>259</td>
</tr>
<tr>
<td>Drag and Drop</td>
<td>261</td>
</tr>
<tr>
<td>Drag and Drop – XAML File</td>
<td>262</td>
</tr>
<tr>
<td>Drag and Drop – Code-Behind</td>
<td>263</td>
</tr>
<tr>
<td>Capturing the Mouse</td>
<td>265</td>
</tr>
<tr>
<td>Bubbled Events</td>
<td>266</td>
</tr>
<tr>
<td>C# Code for Bubbled Event Demo</td>
<td>268</td>
</tr>
<tr>
<td>Lab 8</td>
<td>269</td>
</tr>
<tr>
<td>Summary</td>
<td>270</td>
</tr>
<tr>
<td>Chapter 9: Animation and Media</td>
<td>273</td>
</tr>
<tr>
<td>Manual Animation</td>
<td>275</td>
</tr>
<tr>
<td>Manual Animation Example</td>
<td>276</td>
</tr>
<tr>
<td>Silverlight Animation</td>
<td>277</td>
</tr>
<tr>
<td>Storyboard Methods</td>
<td>278</td>
</tr>
<tr>
<td>User Interface</td>
<td>279</td>
</tr>
<tr>
<td>AnimatedStretch Example</td>
<td>280</td>
</tr>
<tr>
<td>DoubleAnimation</td>
<td>282</td>
</tr>
<tr>
<td>ColorAnimation</td>
<td>283</td>
</tr>
<tr>
<td>ColorAnimation Example</td>
<td>284</td>
</tr>
<tr>
<td>ColorAnimation – Code-Behind</td>
<td>285</td>
</tr>
<tr>
<td>PointAnimation</td>
<td>286</td>
</tr>
<tr>
<td>Audio and Video</td>
<td>287</td>
</tr>
<tr>
<td>MediaElement Example</td>
<td>288</td>
</tr>
<tr>
<td>Resources</td>
<td>290</td>
</tr>
<tr>
<td>Loose Files as Resources</td>
<td>291</td>
</tr>
<tr>
<td>Lab 9</td>
<td>292</td>
</tr>
<tr>
<td>Summary</td>
<td>293</td>
</tr>
<tr>
<td>Chapter 10: Networking</td>
<td>299</td>
</tr>
<tr>
<td>-------------------------------------------------------------</td>
<td>-----</td>
</tr>
<tr>
<td>Silverlight Application Structure</td>
<td>301</td>
</tr>
<tr>
<td>Application Structure Examples</td>
<td>302</td>
</tr>
<tr>
<td>A Simple Package</td>
<td>303</td>
</tr>
<tr>
<td>Application Package</td>
<td>304</td>
</tr>
<tr>
<td>Application Manifest</td>
<td>305</td>
</tr>
<tr>
<td>Application Class</td>
<td>306</td>
</tr>
<tr>
<td>Library Example</td>
<td>307</td>
</tr>
<tr>
<td>XML Processing</td>
<td>308</td>
</tr>
<tr>
<td>Image Resource Files</td>
<td>309</td>
</tr>
<tr>
<td>Resource File in a Component</td>
<td>310</td>
</tr>
<tr>
<td>Resource File in the Package</td>
<td>311</td>
</tr>
<tr>
<td>Application Library Caching</td>
<td>312</td>
</tr>
<tr>
<td>Cache Example</td>
<td>313</td>
</tr>
<tr>
<td>External Parts</td>
<td>314</td>
</tr>
<tr>
<td>OnDemand Example</td>
<td>315</td>
</tr>
<tr>
<td>CopyLocal is False</td>
<td>316</td>
</tr>
<tr>
<td>Using WebClient</td>
<td>317</td>
</tr>
<tr>
<td>Downloading Content on Demand</td>
<td>318</td>
</tr>
<tr>
<td>DownLoadImage Code</td>
<td>319</td>
</tr>
<tr>
<td>Lab 10A</td>
<td>320</td>
</tr>
<tr>
<td>Accessing Web Services in Silverlight</td>
<td>321</td>
</tr>
<tr>
<td>REST</td>
<td>322</td>
</tr>
<tr>
<td>EchoREST Web Service</td>
<td>323</td>
</tr>
<tr>
<td>Demo: EchoREST</td>
<td>324</td>
</tr>
<tr>
<td>SOAP</td>
<td>329</td>
</tr>
<tr>
<td>Web Service Proxies</td>
<td>330</td>
</tr>
<tr>
<td>Windows Communication Foundation</td>
<td>331</td>
</tr>
<tr>
<td>EchoSOAP Web Service</td>
<td>332</td>
</tr>
<tr>
<td>Demo: EchoSOAP</td>
<td>333</td>
</tr>
<tr>
<td>Lab 10B</td>
<td>338</td>
</tr>
<tr>
<td>Summary</td>
<td>339</td>
</tr>
<tr>
<td>Chapter 11: Data Controls and Data Binding</td>
<td>347</td>
</tr>
<tr>
<td>ListBox Control</td>
<td>349</td>
</tr>
<tr>
<td>BookList Example</td>
<td>350</td>
</tr>
<tr>
<td>Book.cs</td>
<td>351</td>
</tr>
<tr>
<td>Auto-Implemented Properties</td>
<td>352</td>
</tr>
<tr>
<td>Using Items Property</td>
<td>353</td>
</tr>
<tr>
<td>Deleting a Book</td>
<td>354</td>
</tr>
<tr>
<td>Selection</td>
<td>355</td>
</tr>
<tr>
<td>Binding to a Collection</td>
<td>356</td>
</tr>
<tr>
<td>DataPager Control</td>
<td>357</td>
</tr>
<tr>
<td>Demo: Using DataPager</td>
<td>358</td>
</tr>
<tr>
<td>DisplayMode</td>
<td>360</td>
</tr>
<tr>
<td>BookPager Step 2</td>
<td>361</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>----------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Chapter 12: Data Access</td>
<td>387</td>
</tr>
<tr>
<td>Accessing Data on the Server</td>
<td>389</td>
</tr>
<tr>
<td>SmallPub Database</td>
<td>390</td>
</tr>
<tr>
<td>BookDb Silverlight Example</td>
<td>392</td>
</tr>
<tr>
<td>DB Class</td>
<td>394</td>
</tr>
<tr>
<td>Language Integrated Query (LINQ)</td>
<td>395</td>
</tr>
<tr>
<td>LINQ Demo</td>
<td>396</td>
</tr>
<tr>
<td>Object Relational Designer</td>
<td>397</td>
</tr>
<tr>
<td>IntelliSense</td>
<td>399</td>
</tr>
<tr>
<td>Web Service Interface</td>
<td>400</td>
</tr>
<tr>
<td>Client Code</td>
<td>401</td>
</tr>
<tr>
<td>Displaying the Books</td>
<td>402</td>
</tr>
<tr>
<td>SelectionChanged Event</td>
<td>403</td>
</tr>
<tr>
<td>Inserting a Book</td>
<td>404</td>
</tr>
<tr>
<td>Client Code for Insert</td>
<td>405</td>
</tr>
<tr>
<td>Deleting a Book</td>
<td>406</td>
</tr>
<tr>
<td>Client Code for Delete</td>
<td>407</td>
</tr>
<tr>
<td>Updating a Book</td>
<td>408</td>
</tr>
<tr>
<td>Client Code for Update</td>
<td>409</td>
</tr>
<tr>
<td>Lab 12</td>
<td>410</td>
</tr>
<tr>
<td>ADO.NET Data Services</td>
<td>411</td>
</tr>
<tr>
<td>ADO.NET Data Services Demo</td>
<td>412</td>
</tr>
<tr>
<td>Choosing Data Objects</td>
<td>413</td>
</tr>
<tr>
<td>BookModel.edmx</td>
<td>414</td>
</tr>
<tr>
<td>ADO.NET Data Service Template</td>
<td>415</td>
</tr>
</tbody>
</table>
## Contents

- **DataService Class** ................................................................. 416
- A REST-based Web Service ......................................................... 417
- A Console Client .................................................................. 418
- A Silverlight Client ................................................................. 420
- Accessing XML Data .............................................................. 423
- Accessing an XML File ........................................................... 424
- Code for ShowXml Example ...................................................... 425
- Using LINQ to XML ............................................................... 426
- Isolated Storage .................................................................. 427
- Isolated Storage Example ....................................................... 428
- Isolated Storage Store ............................................................ 429
- Saving Data to a File ............................................................. 430
- Loading Data from a File ......................................................... 431
- Deleting a File ..................................................................... 432
- Summary ............................................................................ 433

### Appendix A: Learning Resources

- **Appendix B: Silverlight and the Browser** ........................................ 445
  - HTML Bridge ................................................................... 447
  - HTML Bridge Example ..................................................... 448
  - HTML and JavaScript ...................................................... 449
  - HtmlPage Class ............................................................... 450
  - JavaScript from Managed Code ...................................... 451
  - Managed Code from JavaScript ...................................... 452
  - Out-of-Browser Support .................................................. 453
  - Out-of-Browser Demo ..................................................... 454
  - Configuring Out-of-Browser Support ............................... 455
  - Installing Out-of-Browser .............................................. 457
  - Running Out-of-Browser ................................................ 458
  - Application Manifest ....................................................... 459
  - Summary ....................................................................... 460
Chapter 1

Introduction to Silverlight
Introduction to Silverlight

Objectives

After completing this unit you will be able to:

• Explain the role of Silverlight in creating and delivering rich Internet applications.

• Describe the differences between Silverlight and other client-side web technologies such as AJAX and Flash.

• Download the Silverlight plug-in to run Silverlight Web applications.

• Use the Silverlight SDK to create Silverlight applications.

• Use the Silverlight Control on your web pages to enable the use of Silverlight.

• Use a XAML file to declaratively specify the user interface presented by your Silverlight content.

• Provide JavaScript code to instantiate the plug-in.
What Is Silverlight?

- Microsoft Silverlight is a client-side web technology for creating rich Internet applications incorporating features such as vector graphics, animation and multimedia.

- It is implemented as a browser plug-in, currently available for:
  - Internet Explorer
  - Firefox
  - Safari

- The Silverlight 3.0 plug-in is a small download, less than 5MB.

- Silverlight is also cross-platform, available for both Windows and Mac OS X.
  - Microsoft is also supporting the development by Novell of a version running on Linux, called Moonlight.

- Silverlight 3.0 has two programming models:
  - The JavaScript API for Silverlight, using JavaScript code interpreted in the browser. This was the only API available in Silverlight 1.0.
  - The Managed API for Silverlight, using compiled code in C# or Visual Basic running on the Common Language Runtime (CLR). Also, dynamic languages like Python and Ruby can be run on the Dynamic Language Runtime (DLR).
Silverlight and AJAX

- **AJAX is another important client-side Web technology for delivering rich Internet applications.**
  - The table compares AJAX and Silverlight.

<table>
<thead>
<tr>
<th>AJAX</th>
<th>Silverlight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Does not require a browser plug-in, and thus runs in a wide range of browsers on many platforms.</td>
<td>The Silverlight plug-in currently runs only with Internet Explorer, Firefox, and Safari. Supported platforms are Windows and Mac OS.</td>
</tr>
<tr>
<td>Relies on DHTML, CSS and JavaScript for achieving UI.</td>
<td>Supports a very rich UI model by virtue of its plug-in.</td>
</tr>
<tr>
<td>Programming limited to JavaScript.</td>
<td>JavaScript, C#, Visual Basic, Ruby and Python.</td>
</tr>
<tr>
<td>Rich set of controls.</td>
<td>Controls furnished by Microsoft and others by third parties</td>
</tr>
<tr>
<td>Supports out-of-band communication between client and server.</td>
<td>Supports on-demand downloading of content.</td>
</tr>
</tbody>
</table>

- **It is entirely feasible to use both AJAX and Silverlight on a web page.**
Silverlight and Flash

• For several years a popular option for providing visually rich content on web pages has been Adobe’s Flash.

  – The table compares Flash and Silverlight.

<table>
<thead>
<tr>
<th>Flash</th>
<th>Silverlight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Both a runtime component and</td>
<td>Only a runtime component. Tools include Silverlight SDK, Visual Studio and</td>
</tr>
<tr>
<td>a design tool.</td>
<td>Expression.</td>
</tr>
<tr>
<td>Several years old and most</td>
<td>Newer and so is not as ubiquitous.</td>
</tr>
<tr>
<td>users already have the runtime</td>
<td></td>
</tr>
<tr>
<td>installed.</td>
<td></td>
</tr>
<tr>
<td>Big learning curve for the</td>
<td>Easy to learn programming model.</td>
</tr>
<tr>
<td>design tool.</td>
<td></td>
</tr>
<tr>
<td>Some visual effects that</td>
<td>Supports higher quality video and pressure-sensitive input devices.</td>
</tr>
<tr>
<td>Silverlight lacks.</td>
<td></td>
</tr>
<tr>
<td>Content not very discoverable</td>
<td>Content that is more discoverable by search engines by virtue of use of XML.</td>
</tr>
<tr>
<td>by search engines due to use</td>
<td></td>
</tr>
<tr>
<td>of compiled script.</td>
<td></td>
</tr>
</tbody>
</table>
Installing the Silverlight Plug-In

- A web page that uses Silverlight can display a special “Install Microsoft Silverlight” image telling the user to download and install the plug-in.

- Clicking on the image will bring a file download window for installing Silverlight.

![Image of File Download - Security Warning]

While files from the Internet can be useful, this file type can potentially harm your computer. If you do not trust the source, do not run or save this software. What's the risk?
Silverlight Tools

- The Microsoft Silverlight 3 Tools for Visual Studio 2008 SP1 is an add-on for Visual Studio 2008 SP1 for developing Silverlight 3 applications
  - You may download it from the official Microsoft Silverlight Site:
    
    http://silverlight.net/getstarted/ 
  
    - It contains the Silverlight 3 runtime, the Silverlight 3 SDK, Visual Studio templates, documentation and samples.

- In this course we will use both the SDK and Visual Studio in our Silverlight development.
  - Using the SDK is very simple, requiring only a text editor, which can be Visual Studio.
  
  - With the Visual Studio templates you can create a Silverlight project in Visual Studio, enabling programming in C# or Visual Basic.

- Silverlight applications typically consist of files of the following types:
  - HTML
  - XAML
  - JavaScript
  - C# or Visual Basic
Hello Silverlight

- We introduce Silverlight with a simple “Hello” program created with just the SDK.
  
Allowing Blocked Contact

• When you double-click on Default.html, you may get a security message from your browser.

• Clicking on the yellow bar brings up a context menu permitting you to allow the blocked content.

• You will be presented with a message box providing a final warning. Click Yes to run the active content.
Default.html

- The HTML file contains an <object> tag for embedding the Silverlight content.

  - In Internet Explorer, the Silverlight add-on is an ActiveX control, while in other browsers it is a plug-in. (But this is an implementation detail.)

```html
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Hello Silverlight</title>
</head>
<body>
<div>
  <object type="application/x-silverlight-2"
          id="silverlightControl"
          width="400" height="300">
    <param name="background"
           value="Lightgray"/>
    <param name="source" value="Page.xaml"/>
  </object>
</div>
</body>
</html>
```

- Note that the Silverlight control is sized 400 x 300 pixels, with a light gray background.

- On the next page we’ll see that the content in the control, as specified in the XAML file, has size 300 x 300 pixels. Thus, a strip 100 pixels wide shows the background gray.
<Canvas
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="300" Height="300"
    Background="White">

    <TextBlock Text="Hello Silverlight"
        FontSize="50">
        <TextBlock.Foreground>
            <LinearGradientBrush>
                <GradientStop Color="Red" Offset="0.0" />  
                <GradientStop Color="Green" Offset="1.0" />  
            </LinearGradientBrush>
        </TextBlock.Foreground>
    </TextBlock>

    <Ellipse Height="200" Width="240"
        Canvas.Left="20" Canvas.Top="80"
        Stroke="Black" StrokeThickness="10"
        Fill="LightBlue"/>

</Canvas>
XAML

- In Silverlight user interface elements are created via a declarative language called XAML.
  - XAML stands for Extensible Application Markup Language.
  - XAML is based on XML.
- In our first example, we specify a text block and an ellipse as children of a canvas container.

```xml
<Canvas>
  <TextBlock>
    ...
  </TextBlock>
  <Ellipse ...
</Canvas>
```

- XAML is quite intuitive and is discussed in detail beginning in Chapter 3.
XHTML

• XHTML specifies the HTML 4.01 standard as an XML application.

• XHTML is stricter, leading to greater uniformity in how pages are rendered by different browsers.
  – It also provides a specification allowing the Web to be used by other devices, such as handheld computers and mobile phones.

• Important differences between XHTML and HTML:
  – Include a reference to the XML namespace within the HTML element.
    
    `<html xmlns="http://www.w3.org/1999/xhtml" >`
  
  – Supply all element names in lower case (XML is case sensitive).
  
  – The `<head>` and `<body>` elements **must** be included.
  
  – Tags must always be closed and nested correctly.
  
  – Attribute values must always be denoted by quotation marks (either single quote or double quote is OK).

• The `<object>` is valid in XHTML 1.1.
JavaScript Helper Functions

• The Silverlight SDK provides a JavaScript file *Silverlight.js* that contains helper functions for embedding the Silverlight plug-in in a Web page.
  – `Silverlight.createObject()` generates a suitable `<object>` element appropriate for the user’s browser.
  – `Silverlight.createObjectEx()` has the same functionality with different syntax for specifying the input parameters.

• Using *Silverlight.js* you don’t have to worry about detecting whether the Silverlight plug-in is available on the client computer and what to do if it is not available.
Using Silverlight.js

- Our HTML file will now bring in two JavaScript files.
  - The file Silverlight.js is supplied by Microsoft in the Silverlight SDK.
  - We create the file createSilverlight.js to encapsulate the call to createObject() or createObjectEx().
  - See Chap01\CreateObject. Here is Default.html:

```html
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Hello Silverlight</title>
    <script type="text/javascript"
        src="Silverlight.js"></script>
    <script type="text/javascript"
        src="createSilverlight.js"></script>
</head>
<body>
    <div id="parent">
        <script type="text/javascript">
            createSilverlight();
        </script>
    </div>
    <p>
        Silverlight lets you display both text and graphics as specified in a XAML file.
    </p>
</body>
</html>
```
createSilverlight.js

• The file createSilverlight.js contains the function createSilverlight().

function createSilverlight()
{
    Silverlight.createObject(
        "Page.xaml",
        // Source property value
        document.getElementById("parent"),
        // parent HTML reference
        "mySilverlightPlugin",
        // Unique plug-in ID value
        // properties:
        {
            width:'400',   // Width in pixels
            height:'300',   // Height in pixels
            background:'LightGray',
            version:'2.0'
        },
        // events:
        {}
    );
}
Using createObjectEx()

- An alternative is to use the `createObjectEx()` function.
  - The syntax difference is that `createObjectEx()` takes a single parameter, which is an associative array.
  - Under the hood, `createObjectEx()` calls `createObject()`.
- For an example, see `Chap01\CreateObjectEx`.
  - The only file that is different is `createSilverlight.js`.

```javascript
function createSilverlight()
{
    Silverlight.createObjectEx(
    {
        source: "Page.xaml",
        parentElement: document.getElementById("parent"),
        id: "mySilverlightPlugin",
        properties: {
            width:'400', height:'300',
            background:'LightGray', version:'2.0'
        },
        events: {}
    }
    );
}
```
Width and Height in Per Cent

- In calling `createObject()` or `createObjectEx()` you may specify the width and height as a percentage as an alternative to pixels.

```javascript
width: '80%', // Width in per cent
height: '70%', // Height in per cent
```

- The plug-in content will then dynamically resize as the browser window is resized.
  
  - See Chap01\RelativeSize for an example.
Providing for User Plug-In Download

- Since a user needs the Silverlight plug-in to view the Silverlight content of your web pages, you should provide a means to make that possible.

- One way is to provide alternate content for the <object> content.
  - For an example, see Chap01\Download.

```xml
<object type="application/x-silverlight-2"
       id="silverlightControl"
       width="400" height="300">
  <param name="background" value="Lightgray"/>
  <param name="source" value="Page.xaml"/>
  <!-- Alternative content -->
  This page requires Microsoft Silverlight.
  <a href="http://www.microsoft.com/silverlight/downloads.aspx">
    Download it.</a>
</object>
```

- Here is what a user without Silverlight installed will see:
Automatic Plug-In Download

- Using the Silverlight.js JavaScript file provides a better solution automatically.

- The `createObject()` and `createObjectEx()` functions will handle the task of displaying alternate content for you.
  - The alternate content is an attractive image that the user can click on to download Silverlight.

- If the CreateObject example is opened by a browser where Silverlight is not installed, this will be displayed:
Summary

- Microsoft Silverlight is a client-side web technology for creating rich Internet applications.

- Silverlight is an alternative to Adobe’s Flash and can be used in conjunction with AJAX.

- A user can download the Silverlight plug-in to run Silverlight Web applications.

- A developer can use the Silverlight SDK to create Silverlight applications.

- The Silverlight Control on a Web page enables the use of Silverlight.

- You can use XAML files to declaratively specify the user interface presented by your Silverlight content.

- JavaScript code can be used to instantiate the plug-in, shielding the developer from issues such as browser differences and detecting the Silverlight plug-in on the client computer and what to do if it is not available.
Chapter 5

Text and Fonts
Text and Fonts

Objectives

After completing this unit you will be able to:

• Display text using the TextBlock element.

• Customize text using fonts and additional properties of TextBlock.

• Create rich text content using the Run and LineBreak elements.
TextBlock

- Silverlight provides the **TextBlock** element for the display of text content.

- You may specify the content using the **Text** property.

  ```xml
  <TextBlock Text="Simple Text Example" />
  ```

- Alternatively, you may use content element syntax, nesting the content between the object tags.

  ```xml
  <TextBlock>
      Simple Text Example
  </TextBlock>
  ```

- To display a second TextBlock without overwriting the first TextBlock, you should use the Canvas.Top attached property.

  - For a small complete example, see **SimpleText**.

    ```xml
    <TextBlock Text="Simple Text Example" />
    <TextBlock Canvas.Top="40">
      A second line of text, using Canvas.Top
    </TextBlock>
    ```

    - Here is the output:

      Simple Text Example

      A second line of text, using Canvas.Top
Text Wrapping

• *TextBlock* has a *TextWrapping* property.
  
  – *NoWrap* (the default behavior) means that no wrapping occurs when text reaches the edge of the containing box.
  
  – *Wrap* means that line breaking occurs when text reaches the edge of the containing box.

• To ensure desired wrapping behavior occurs, you should explicitly set the *Width* property.
  
  – See *TextWrapping* for a complete example.

```xml
<TextBlock Text="Text Wrapping Example" />
<TextBlock Canvas.Top="40" Width="400"
            TextWrapping="Wrap">
  Our next example consists of two TextBlock elements. The first TextBlock uses the Text property, and the second TextBlock uses content element syntax. The Canvas.Top attached property is used to position the second TextBlock below the first TextBlock. We use the default font and specify wrapping.
</TextBlock>
```

– Here is the output:

```
Text Wrapping Example

Our next example consists of two TextBlock elements. The first TextBlock uses the Text property, and the second TextBlock uses content element syntax. The Canvas.Top attached property is used to position the second TextBlock below the first TextBlock. We use the default font and specify wrapping.
```
## Font Attributes

- The table shows the various font attributes that can be specified as properties of *TextBlock*.

<table>
<thead>
<tr>
<th>FontFamily</th>
<th>Set of typefaces that share the same family name but can differ in features such as font size</th>
</tr>
</thead>
<tbody>
<tr>
<td>FontSize</td>
<td>Size of font in pixels. Default value is 14.666 pixels.</td>
</tr>
<tr>
<td>FontStretch</td>
<td>Degree to which a font form is stretched from its normal aspect ratio</td>
</tr>
<tr>
<td>FontStyle</td>
<td>Whether the font style is normal or italic</td>
</tr>
<tr>
<td>FontWeight</td>
<td>Relative weight of a font in terms of the lightness or heaviness of its strokes</td>
</tr>
</tbody>
</table>
Font Families

• There are nine different font families packaged with Silverlight.

• The default font is *Lucida Sans Unicode, Lucida Grande*.
  
  – *Lucida Sans Unicode* is included with Windows.
  
  – *Lucida Grande* is an almost identical font included with Mac OS.
  
  – An alias for this default font is *Portable User Interface*.

• An example program displays the name of each font family, using the font in the display.
  
  – See *FontFamilies*.

  Arial
  **Arial Black**
  Comic Sans MS
  Courier New
  Georgia
  Lucida Sans Unicode, Lucida Grande
  Times New Roman
  Trebuchet MS
  Verdana
Font Weights

- *FontWeight* specifies the relative weight of the font—the lightness or darkness of its strokes.

- The default font weight is *Normal*, and its main alternative is *Bold*.

- The *FontWeight* property can be set to ten different values, but the built-in fonts support only two weights.

- An example program displays the 10 values for *FontWeight*, using the weight in the display.
  
  - See *FontWeights*.

  Thin
  ExtraLight
  Light
  Normal
  Medium
  SemiBold
  Bold
  ExtraBold
  Black
  ExtraBlack
**Additional TextBlock Properties**

- Besides *TextWrapping* and the various font attributes, there are two other properties of *TextBlock* that can control the appearance of text.

- *TextDecorations* controls underlining of text.
  - Default value is *None*.
  - The value *Underline* indicates that the text should be underlined.

- *Foreground* specifies a brush that is used for rendering the text.
  - In particular, using the *Foreground* property you can specify the color of text.
  - In the next chapter we’ll look at various kinds of brushes that can be used for painting both shapes and text.
TextBlock Dimensions

- Sometimes you need to retrieve the dimensions of a TextBlock.
  - You can do this via the ActualWidth and ActualHeight properties.

- As an example, let’s draw a Rectangle border about a TextBlock.
  - The example program is TextBorder.
  - As our first attempt, we guess at the Width and Height, making the Width the same as the Width of the TextBlock.

```xml
<TextBlock Width="100"
    Canvas.Left="50" Canvas.Top="50"
    FontSize="20" TextWrapping="Wrap"
    x:Name="tb">
    Here is some sample text
</TextBlock>

<Rectangle Width="100" Height="110"
    Canvas.Left="50" Canvas.Top="50"
    Stroke="Black" x:Name="rect"/>

- Here is the result when we run it:
ActualWidth and ActualHeight

- In the code-behind file we assign the dimensions of the Rectangle to be the ActualWidth and ActualHeight of the TextBlock.

```csharp
private void Canvas_Loaded(object sender, RoutedEventArgs e)
{
    rect.Width = tb.ActualWidth;
    rect.Height = tb.ActualHeight;
}
```

- Here is the result when we run the program:
Rich Text Content

- It’s quite feasible to obtain quite rich text content in a TextBlock, with various formats applied to different parts of the text content.
  - See RichText for an example of such rich text content.
Run and LineBreak

- The content property of TextBlock is a collection called Inlines, which consists of Run and LineBreak elements.
  - XAML uses implicit collection syntax for this collection, omitting a tag for Inlines.
  - Here is the XAML for our example:

```xml
<TextBlock>
  <Run FontSize="24" FontFamily="Verdana"
       FontWeight="Bold">Rich Text</Run>
  <LineBreak/>
  <Run>A line of ordinary text with default font</Run>
  <LineBreak/>
  <Run FontFamily="Times New Roman">Times New Roman with</Run>
  <LineBreak/>
  <Run FontFamily="Times New Roman" FontStyle="Italic">italics for emphasis</Run>
  <LineBreak/>
  <Run FontFamily="Verdana">Verdana with</Run>
  <LineBreak/>
  <Run FontFamily="Verdana" FontWeight="Bold">bold for emphasis</Run>
  <LineBreak/>
  <Run FontFamily="Arial">Following line uses</Run>
  <Run FontFamily="Arial" TextDecorations="Underline">Courier New</Run>
  <LineBreak/>
  <Run FontFamily="Arial">for program text:</Run>
  <LineBreak/>
  <Run FontFamily="Courier New" Text="&lt;Run&gt;Hi, there!&lt;/Run&gt;">&lt;Run&gt;Hi, there!&lt;/Run&gt;</Run>
  <LineBreak/>
  ...
</TextBlock>
```
Lab 5

Dynamic Text Demonstration

In this lab, you will implement a program to dynamically update the properties of a TextBlock. The lab provides more practice with XAML and C# programming and will also let you experiment with various TextBlock properties.

Here is some other text

Top: 30  Left: 70  Width: 100
Text: Here is some other text
TextWrapping: Wrap
Show Border: No
FontSize: 24  Foreground: Blue
FontFamily: Comic Sans MS
FontStyle: Normal
FontWeight: Normal
TextDecorations: Underline

Detailed instructions are contained in the Lab 5 write-up at the end of the chapter.

Suggested time: 45 minutes
Summary

- Silverlight provides the `TextBlock` element for displaying text.

- You can customize text using fonts and additional properties of `TextBlock` such as `TextWrapping`, `TextDecorations`, and `Foreground`.

- You can create rich text content using the `Run` and `LineBreak` elements.
Lab 5

Dynamic Text Demonstration

Introduction

In this lab, you will implement a program to dynamically update the properties of a TextBlock. The lab provides more practice with XAML and C# programming and will also let you experiment with various TextBlock properties.

Suggested Time: 45 minutes

Root Directory: OIC\Silver Cs

Directories: Labs\Lab5\DynamicText (do your work here)
Chap05\DynamicText\Step1 (backup of starter code)
Chap05\DynamicText\Step2 (answer)

Part 1. Examine Starter Code

1. Build and run the starter project. You will see the phrase “This is some text” displayed at the top left.

2. Click the Refresh button. This will load some new text at the location specified by Top and Left. You can also specify wrap or no wrap and whether to show a border around the text.

Here is some other text
3. Observe that no text is truncated. What happens if you set TextWrapping to Wrap and click Refresh again? To help in visualizing clearly the extent of the TextBlock, set Show Border to Yes.

4. You will see additional text wrap to the next line. Now Width does influence how the TextBlock is shown. But no word is split in the middle. Try setting the Width to 50.

5. Now examine the code in the code-behind file.

   a. The combo boxes are initialized, and a Rectangle is created to use as a border to the TextBlock.

```csharp
private Rectangle rect;

private void LayoutRoot_Loaded(object sender, RoutedEventArgs e)
{
    cmbTextWrapping.Items.Add("NoWrap");
    cmbTextWrapping.Items.Add("Wrap");
    cmbTextWrapping.SelectedIndex = 0;

    cmbShowBorder.Items.Add("No");
    cmbShowBorder.Items.Add("Yes");
    cmbShowBorder.SelectedIndex = 0;
```
b. The `onRefresh()` event handler sets the various attributes based on the information the user entered into the controls.

```csharp
private void onRefresh(object sender, RoutedEventArgs e)
{
    double top = Convert.ToDouble(txtTop.Text);
    double left = Convert.ToDouble(txtLeft.Text);
    Canvas.SetTop(tbMessage, top);
    Canvas.SetLeft(tbMessage, left);
    tbMessage.Width = Convert.ToDouble(txtWidth.Text);
    tbMessage.Text = txtText.Text;

    string strWrap = cmbTextWrapping.SelectedItem.ToString();
    if (strWrap == "NoWrap")
        tbMessage.TextWrapping = TextWrapping.NoWrap;
    else if (strWrap == "Wrap")
        tbMessage.TextWrapping = TextWrapping.Wrap;

    Canvas.SetTop(rect, top);
    Canvas.SetLeft(rect, left);
    rect.Width = tbMessage.ActualWidth;
    rect.Height = tbMessage.ActualHeight;

    string strBorder = cmbShowBorder.SelectedItem.ToString();
    if (strBorder == "Yes")
        rect.StrokeThickness = 1;
    else
        rect.StrokeThickness = 0;
}
```

### Part 2. Additional Customizations

1. Add a StackPanel containing text boxes for specifying FontSize and Foreground. Here is how the UI should appear:

![Screenshot of UI with FontSize: 24 and Foreground: Blue]

2. Add code to `onRefresh()` to set these properties to the values entered by the user.

```csharp
tbMessage.FontSize = Convert.ToDouble(txtFontSize.Text);
tbMessage.Foreground = BrushFromString(txtForeground.Text);
```

3. Test the new features. Note that with NoWrap, the text in the TextBlock is not even truncated by the canvas. It would only be truncated by the Silverlight control, and the
Visual Studio generated test page by default makes the Silverlight control fill the window of the browser.

Here is some other text

Top: 30  Left: 70  Width: 100
Text: Here is some other text
TextWrapping: NoWrap
Show Border: No
FontSize: 24  Foreground: Blue

4. Add XAML for a combo box for choosing a font family.

```xml
<StackPanel Orientation="Horizontal">
  <TextBlock Margin="5,9,5,0">FontFamily:</TextBlock>
  <ComboBox Margin ="0,5,0,0" Width="175"
      x:Name="cmbFontFamily">
  </ComboBox>
</StackPanel>
```

5. Add code to `LayoutRoot_Loaded()` to initialize this combo box.

```csharp
cmbFontFamily.Items.Add("Portable User Interface");
cmbFontFamily.Items.Add("Arial");
cmbFontFamily.Items.Add("ArialBlack");
cmbFontFamily.Items.Add("Comic Sans MS");
cmbFontFamily.Items.Add("Courier New");
cmbFontFamily.Items.Add("Lucida Grande");
cmbFontFamily.Items.Add("Lucida Sans Unicode");
cmbFontFamily.Items.Add("Times New Roman");
cmbFontFamily.Items.Add("Trebuchet MS");
cmbFontFamily.Items.Add("Verdana");
cmbFontFamily.SelectedIndex = 0;
```

6. Add code to `onRefresh()` to set the `FontFamily` property to the value entered by the user.

```csharp
tbMessage.FontFamily =
    new FontFamily(cmbFontFamily.SelectedItem.ToString());
```
7. Test the new feature.

8. Now let’s complete the program by adding XAML for combo boxes for choosing FontStyle, FontWeight and TextDecorations.

```
<StackPanel Orientation="Horizontal">
  <TextBlock Margin="5,9,5,0">FontStyle:</TextBlock>
  <ComboBox Margin="0,5,0,0" Width="100"
    x:Name="cmbFontStyle">
  </ComboBox>
</StackPanel>

<StackPanel Orientation="Horizontal">
  <TextBlock Margin="5,9,5,0">FontWeight:</TextBlock>
  <ComboBox Margin="0,5,0,0" Width="100"
    x:Name="cmbFontWeight">
  </ComboBox>
</StackPanel>

<StackPanel Orientation="Horizontal">
  <TextBlock Margin="5,9,5,0">TextDecorations:</TextBlock>
  <ComboBox Margin="0,5,0,0" Width="100"
    x:Name="cmbTextDecorations">
  </ComboBox>
</StackPanel>
```

9. Add code to LayoutRoot_Loaded() to initialize these combo boxes.

```
cmbFontStyle.Items.Add("Normal");
cmbFontStyle.Items.Add("Italic");
cmbFontStyle.SelectedIndex = 0;

cmbFontWeight.Items.Add("Thin");
cmbFontWeight.Items.Add("Normal");
cmbFontWeight.Items.Add("Bold");
cmbFontWeight.Items.Add("ExtraBlack");
cmbFontWeight.SelectedIndex = 1;

cmbTextDecorations.Items.Add("None");
cmbTextDecorations.Items.Add("Underline");
cmbTextDecorations.SelectedIndex = 0;
```

10. Add code to onRefresh() to set these properties to the values entered by the user.

```
string strFontStyle = cmbFontStyle.SelectedItem.ToString();
if (strFontStyle == "Italic")
  tbMessage.FontStyle = FontStyles.Italic;
else
  tbMessage.FontStyle = FontStyles.Normal;

string strFontWeight = cmbFontWeight.SelectedItem.ToString();
if (strFontWeight == "Thin")
  tbMessage.FontWeight = FontWeights.Thin;
else if (strFontWeight == "Normal")
  tbMessage.FontWeight = FontWeights.Normal;
else if (strFontWeight == "Bold")
  tbMessage.FontWeight = FontWeights.Bold;
```
else if (strFontWeight == "ExtraBlack")
    tbMessage.FontWeight = FontWeights.ExtraBlack;

string strTextDecorations = cmbTextDecorations.SelectedItem.ToString();
if (strTextDecorations == "Underline")
    tbMessage.TextDecorations = TextDecorations.Underline;
else
    tbMessage.TextDecorations = null;

11. Test the new features.

12. Now spend a few minutes checking out all the various features of the program, experimenting with the various TextBlock properties that you can set.
Chapter 8

Handling Events
Handling Events

Objectives

After completing this unit you will be able to:

- Specify event handlers in XAML.
- Add and remove event handlers in C# code.
- Handle mouse events in your Silverlight applications.
- Perform rubber-band drawing and drag and drop.
- Explain event bubbling.
Events in Silverlight

• There are two general kinds of events supported by Silverlight:
  – Input events
  – Non-input events

• Input events pertain to some action by the user, such as use of the mouse or keyboard.
  – Input stimuli are handled first by the browser that hosts the Silverlight plug-in and then are dispatched to Silverlight.

• Non-input events typically report a state change, such as the plug-in or a UI element being loaded.

• Most events are handled on XAML elements.
Event Handlers in XAML

- Most events pertain to XAML elements and can be specified directly in XAML.

- As an example we illustrate handling the `MouseLeftButtonDown` for a XAML rectangle.
  - See SimpleEvent.
Simple Event Example

Here is the XAML:

```xml
<Rectangle Width="200" Height="100"
    Fill="Red"
    MouseLeftButtonDown="onButtonDown"
    x:Name="myRect"/>
```

And here is the C# for the handler:

```csharp
private void onButtonDown(object sender, MouseButtonEventArgs e)
{
    string msg = "onButtonDown\n";
    Rectangle rect = sender as Rectangle;
    msg += "sender = " + rect.Name + "\n";
    double x = e.GetPosition(myRect).X;
    msg += "X = " + x + "\n";
    double y = e.GetPosition(myRect).Y;
    msg += "Y = " + y + "\n";
    MessageBox.Show(msg, "SimpleEvent",
                    MessageBoxButton.OK);
}
```

− The first parameter `sender` refers to the object that is the source of the event.

− The second parameter identifies event data for a specific event. For example, a mouse event argument has a `GetPosition()` method for finding the X and Y coordinates of the mouse position.
Adding/Removing Event Handlers

- You may dynamically add an event handler using delegate notation, in the same manner as in other .NET coding.

```csharp
private void onButtonDown(object sender, MouseButtonEventArgs e)
{
    ...
}

rect.MouseLeftButtonDown += onButtonDown;
```

- In a similar way you may dynamically remove an event handler

```csharp
rect.MouseLeftButtonDown -= onButtonDown;
```
Dynamic Event Handling Example

- Our second example illustrates event handling for two rectangles.
  - See AddRemove\Step1.

  - There is a handler for the `MouseLeftButtonDown` event on the second rectangle but not on the first.

- But the XAML shows the reverse!

```xml
<Rectangle
    Width="100" Height="50" Fill="Red"
    MouseLeftButtonDown="onButtonDown"
    Loaded="onLoaded"
    x:Name="rect1"/>

<Rectangle
    Width="100" Height="50" Fill="Green"
    Canvas.Left="50" Canvas.Top="75"
    Loaded="onLoaded"
    x:Name="rect2"/>
```
Loaded Event

- The key to understanding this example is the fact that each UI child element of the canvas handles the Loaded event.

```xml
<Rectangle
    Width="100" Height="50" Fill="Red"
    MouseLeftButtonDown="onButtonDown"
    Loaded="onLoaded"
    x:Name="rect1"/>

<Rectangle
    Width="100" Height="50" Fill="Green"
    Canvas.Left="50" Canvas.Top="75"
    Loaded="onLoaded"
    x:Name="rect2"/>
```

- The handler for the Loaded event removes the handler for the first rectangle and adds a handler for the second rectangle.

```csharp
private void onLoaded(object sender, RoutedEventArgs e)
{
    Rectangle rect = sender as Rectangle;
    if (rect.Name == "rect1")
        rect.MouseLeftButtonDown -= onButtonDown;
    else if (rect.Name == "rect2")
        rect.MouseLeftButtonDown += onButtonDown;
}
```
Mouse Events

- **Silverlight defines five different mouse events.**

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MouseLeftButtonDown</td>
<td>Raised when the left mouse button is pressed</td>
</tr>
<tr>
<td>MouseLeftButtonUp</td>
<td>Raised when the left mouse button is released</td>
</tr>
<tr>
<td>MouseMove</td>
<td>Raised when the coordinate position of the mouse pointer changes</td>
</tr>
<tr>
<td>MouseEnter</td>
<td>Raised when the mouse enters the bounding area of an object</td>
</tr>
<tr>
<td>Mouse Leave</td>
<td>Raised when the mouse leaves the bounding area of an object</td>
</tr>
</tbody>
</table>

- **Silverlight does not define any right mouse button events.**
  - Right-clicking on the Silverlight control brings up a context menu for configuring the Silverlight add-on.
Microsoft Silverlight Configuration

- Selecting the Silverlight menu from the right-click operation brings up a dialog for configuring Silverlight.

![Microsoft Silverlight Configuration dialog]

Choose how Silverlight installs updates

Silverlight can periodically check for, download and install updates automatically using these settings.

- **Install updates automatically (recommended)**
- Check for updates, but let me choose whether to download and install them
- **Never check for updates**
  
  Your computer may become vulnerable to security threats. Use this setting only if you receive Silverlight updates from Microsoft Update or another source.

[Silverlight Privacy Statement]
MouseEventArgs

- Handlers for mouse events take the usual two arguments.

- The first parameter sender refers to the object that is the source of the event.

- The second parameter is of type MouseEventArgs or a derived class such as MouseButtonEventArgs and provides mouse-specific event information.
  - The method GetPosition(element) returns a Point with properties X and Y giving the coordinates.
  - The coordinates are relative to element. If element is null, the coordinates are relative to the Silverlight control.
Mouse Position Example

• The example program shows the coordinates of the mouse when the `LeftMouseButtonUp` event is raised.
  
  − See `MousePosition`.

• The XAML specifies a blank canvas, which will be dynamically populated with child elements.

```xml
<Canvas
  Background="SeaShell"
  MouseLeftButtonUp="addChild"
  x:Name="can">
</Canvas>
```
private void addChild(object sender,
    MouseEventArgs e)
{
    double x = e.GetPosition(can).X;
    double y = e.GetPosition(can).Y;
    string strPt = "(" + x + ", " + y + ")";
    string xamlFragment = "<TextBlock" + " xmlns='http://schemas.microsoft.com/client/2007'>" + strPt + "</TextBlock>";
    Object obj = XamlReader.Load(xamlFragment);
    TextBlock tb = obj as TextBlock;
    can.Children.Add(tb);
    Canvas.SetLeft(tb, x);
    Canvas.SetTop(tb, y);
}
Rubber-Band Drawing

- Our next example program illustrates “rubber-band” drawing.
  - See DrawRectangles\Step2.
  - When the left mouse button is pressed and the mouse is moved around, the outline of a rectangle is shown.
  - When the button is released, the rectangle is permanently added as a child of the canvas, with a blue fill.
Naïve Implementation

- A naïve implementation that “almost” works results in a whole family of rubber-band outline rectangles.
  
  - See DrawRectangles\Step1

![Diagram of a blank canvas with handlers for MouseLeftButtonDown, MouseMove, and MouseLeftButtonUp.]

- The XAML specifies a blank canvas with handlers for MouseLeftButtonDown, MouseMove, and MouseLeftButtonUp.

```xml
<Canvas
  Background="SeaShell"
  Name="can"
  MouseLeftButtonDown="onButtonDown"
  MouseMove="onMouseMove"
  MouseLeftButtonUp="onButtonUp">
</Canvas>
```
Naïve Implementation (Cont’d)

- The code-behind file defines some member variables

```csharp
double anchorX, anchorY;
bool track;
```

- The button down handler establishes an anchor and sets the `track` variable to `true`.

```csharp
private void onButtonDown(object sender, MouseButtonEventArgs e)
{
    anchorX = e.GetPosition(can).X;
    anchorY = e.GetPosition(can).Y;
    track = true;
}
```

- The `MouseMove` handler draws a tracking rectangle from the anchor to the current mouse position.

```csharp
private void onMouseMove(object sender, MouseEventArgs e)
{
    if (!track)
        return;
    double x = e.GetPosition(can).X;
    double y = e.GetPosition(can).Y;
    drawRect(x, y);
}
```
Naïve Implementation (Cont’d)

- The button up handler terminates tracking and draws the final rectangle.

```csharp
private void onButtonUp(object sender, MouseEventArgs e)
{
    track = false;
    double x = e.GetPosition(can).X;
    double y = e.GetPosition(can).Y;
    drawRect(x, y);
}
```
drawRect() Helper Method

- The **drawRect()** helper function draws a rectangle from the anchor to the current mouse position. It is a dashed outline rectangle if we are tracking, otherwise it will be a blue filled rectangle.

```csharp
private void drawRect(double x, double y)
{
    string xamlFragment = "<Rectangle" +
    " xmlns='http://schemas.microsoft.com/client/2007' />";
    Object obj = XamlReader.Load(xamlFragment);
    Rectangle rect = obj as Rectangle;
    can.Children.Add(rect);
    Canvas.SetLeft(rect, Math.Min( anchorX, x));
    Canvas.SetTop( rect, Math.Min( anchorY, y));
    rect.Width = Math.Abs(x - anchorX);
    rect.Height = Math.Abs(y - anchorY);
    if (track)
    {
        rect.Stroke = BrushFromString("Black");
        rect.StrokeThickness = 1;
        DoubleCollection dc = new DoubleCollection();
        dc.Add(4);
        dc.Add(2);
        rect.StrokeDashArray = dc;
    } else
    {
        rect.Fill = BrushFromString("Blue");
    }
}
```
Final Implementation

• The final implementation needs to add only a little code (see Step 2).
  
  – Declare a member variable `trackRect` that will store the tracking rectangle element.

```csharp
Rectangle trackRect;

  – This variable is initialized when the tracking element is initialized.

if (track)
{
    rect.Stroke = BrushFromString("Black");
    rect.StrokeThickness = 1;
    DoubleCollection dc = new DoubleCollection();
    dc.Add(4);
    dc.Add(2);
    rect.StrokeDashArray = dc;
    trackRect = rect;
}

  – At the end of `drawRect()`, if we are not tracking set the tracking element to null.

else
{
    rect.Fill = BrushFromString("Blue");
    trackRect = null;
}
```
Final Implementation (Cont’d)

- Remove the tracking element from the children collection at the beginning of `drawRect()`.

```csharp
private void drawRect(double x, double y)
{
    // If there is a tracking rectangle, remove it
    if (trackRect != null)
        can.Children.Remove(trackRect);
    ...
```

Drag and Drop

- Similar logic can be used to implement drag and drop.
  - See DragDrop for an example.
  - Note that as you drag a rectangle, the Z-order is preserved, and so a rectangle may be dragged “under” another rectangle.
Drag and Drop – XAML File

- The XAML file defines handlers for `MouseLeftButtonDown`, `MouseMove`, and `MouseLeftButtonUp` for each of the three rectangles.

```xml
<Canvas
    Background="SeaShell"
    x:Name="can">
    <Rectangle Height="60" Width="100" Fill ="Red"
        MouseLeftButtonDown="onMouseDown"
        MouseMove="onMouseMove"
        MouseLeftButtonUp="onMouseUp"
    />
    <Rectangle Height="60" Width="100"
        Fill ="Green"
        Canvas.Left="50" Canvas.Top="40"
        MouseLeftButtonDown="onMouseDown"
        MouseMove="onMouseMove"
        MouseLeftButtonUp="onMouseUp"
    />
    <Rectangle Height="60" Width="100"
        Fill ="Blue"
        Canvas.Left="100" Canvas.Top="80"
        MouseLeftButtonDown="onMouseDown"
        MouseMove="onMouseMove"
        MouseLeftButtonUp="onMouseUp"
    />
</Canvas>
```
Drag and Drop – Code-Behind

bool dragging = false;

// Start drag and drop
private void onMouseDown(object sender,
    MouseButtonEventArgs e)
{
    startX = e.GetPosition(can).X;
    startY = e.GetPosition(can).Y;

    dragging = true;

    // Ensure that this object is the only one
    // receiving mouse events
    Rectangle rect = sender as Rectangle;
    rect.CaptureMouse();
}
private void onMouseMove(object sender, MouseEventArgs e)
{
    if (dragging == true)
    {
        var currX = e.GetPosition(can).X;
        var currY = e.GetPosition(can).Y;

        // Reset the location of the object
        Rectangle rect = sender as Rectangle;
        double newLeft = Canvas.GetLeft(rect)
                        + currX - startX;
        double newTop = Canvas.GetTop(rect)
                        + currY - startY;
        Canvas.SetLeft(rect, newLeft);
        Canvas.SetTop(rect, newTop);

        // Update the start position of the mouse
        startX = currX;
        startY = currY;
    }
}

// Stop drag and drop
private void onMouseUp(object sender, MouseButtonEventArgs e)
{
    dragging = false;

    // Allow all objects to receive mouse events
    Rectangle rect = sender as Rectangle;
    rect.ReleaseMouseCapture();
}
Capturing the Mouse

• There is a potential problem in drag and drop operations.
  – When you have pressed the mouse button down and are dragging, you might drag the mouse pointer over another object, which could start to receive the mouse messages.

• You can prevent this problem by capturing the mouse.
  – In the handler for `MouseLeftButtonDown`, call the method `CaptureMouse()`, which is defined on UI elements such as `Rectangle`. The element capturing the mouse will then receive all mouse messages until the capture is released.

    // Ensure that this object is the only one receiving mouse events
    Rectangle rect = sender as Rectangle;
    rect.CaptureMouse();

  – In the handler for `MouseLeftButtonUp`, call the method `ReleaseMouseCapture()`. The mouse event behavior will then return to normal.

    // Allow all objects to receive mouse events
    Rectangle rect = sender as Rectangle;
    rect.ReleaseMouseCapture();
Bubbled Events

- Mouse events in Silverlight are *bubbled*, which means they are passed from a child object and are forwarded up to each of the successive parent elements in the XAML object hierarchy.
  
  - For an example, see BubbleEvents.
  
  - The XAML hierarchy has a Canvas with a nested Canvas and then a Rectangle.
  
  - There is a handler of **MouseLeftButtonDown** for each element in the hierarchy.

```xml
<Canvas
    Height="200" Width="400"
    Background="LightBlue"
    Margin="10"
    MouseLeftButtonDown="onMouseDown"
    x:Name="can1">

    <Canvas Canvas.Top="20"
        Height="100" Width="150"
        Background="Yellow"
        MouseLeftButtonDown="onMouseDown"
        x:Name="can2">

        <Rectangle Height="40" Width="100"
            Canvas.Top="40"
            Fill="Blue"
            MouseLeftButtonDown="onMouseDown"
            x:Name="rect"/>

    </Canvas>
</Canvas>
```
Bubbled Events (Cont’d)

- If you click on the inside rectangle, you will see the mouse message forwarded to the parent canvas and then to the root canvas.
  - Run the program on the debugger

- Here is the result of `Debug.WriteLine()` statements in the mouse button down event handler. You’ll find this output in Visual Studio’s Output window.

```
rect MouseDown
can2 MouseDown
can1 MouseDown
```
C# Code for Bubbled Event Demo

• Here is the code:

  – The only tricky aspect is checking for the type of the sender. The **Name** property is present for **Canvas** and **Rectangle** but not for an arbitrary **object**.

```csharp
using System.Diagnostics;
...

private void onMouseDown(object sender, MouseButtonEventArgs e)
{
    string msg = ""
    if (sender.GetType() == typeof(Canvas))
    {
        Canvas can = sender as Canvas;
        msg = can.Name + " MouseDown";
    }
    else if (sender.GetType() == typeof(Rectangle))
    {
        Rectangle rec = sender as Rectangle;
        msg = rec.Name + " MouseDown";
    }
    Debug.WriteLine(msg);
}
```

  – Also, we must import the **System.Diagnostics** namespace in order to use **Debug.WriteLine()**.

  – If you can’t run the program in the Visual Studio debugger, you may have reinstalled the Silverlight 3.0 plug-in after installing the Silverlight 3.0 Tools for Visual Studio 2008. You should be able to fix this problem by doing a reinstall of the tools.
Lab 8

Bubbled Events

In this lab, you will modify the AddRemove program to illustrate event bubbling. You’ll handle the mouse button down event for the Canvas and observe the behavior when you click the mouse at various locations.

Detailed instructions are contained in the Lab 8 write-up at the end of the chapter.

Suggested time: 20 minutes
Summary

- Event handlers can be specified in XAML.
- You can add and remove event handlers in C# code.
- By handling mouse events you can perform rubber-band drawing and drag and drop in your Silverlight applications.
- Mouse events are bubbled, passed up the XAML object hierarchy.
Lab 8

Bubbled Events

Introduction

In this lab, you will modify the AddRemove program to illustrate event bubbling. You’ll handle the mouse button down event for the Canvas and observe the behavior when you click the mouse at various locations.

Suggested Time: 20 minutes

Root Directory: OIC\SilverCs

Directories: Labs\Lab8\AddRemove (do your work here) Chap08\AddRemove\Step1 (backup of starter code) Chap08\AddRemove\Step2 (answer)

Instructions

1. Build and run the starter project. Review the code.

2. In the XAML file add a handler of the MouseLeftButtonDown event in the canvas.

   <Canvas Background="LightBlue"
          MouseLeftButtonDown="Canvas_MouseLeftButtonDown">

3. Implement this handler to initially do the following:

   a. Create a message string that shows the name of the event handler.

   b. If the sender was the Canvas, append to the string “Canvas”, and if the sender was a rectangle, append the name of the particular rectangle that sent the message.

   c. Display the message in a message box.

   ```csharp
   private void Canvas_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)
   {
       string msg = "Canvas_MouseLeftButtonDown\n";
       UIElement elt = sender as UIElement;
       string name = "";
       if (elt.GetType() == typeof(Canvas))
         name = "Canvas";
       else if (elt.GetType() == typeof(Rectangle))
       { Rectangle rect = elt as Rectangle;
   ```
name = rect.Name;
}
msg += "sender = " + name + "\n";  
MessageBox.Show(msg, "SimpleEvent", MessageBoxButton.OK);
}

4. Build and run. Observe what happens when you click over each rectangle and over an area of the canvas where there is no rectangle. Do you observe the event bubbling when you click over the second rectangle?

5. At present we only display the mouse coordinates in the handler for clicking over a rectangle. Add code to display the mouse coordinates in the handler for clicking over the canvas. This time, though, make the mouse coordinates relative to the Silverlight control rather than relative to the canvas.

private void Canvas_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)
{
...
msg += "sender = " + name + "\n";
double x = e.GetPosition(null).X;
msg += "X = " + x + "\n";
double y = e.GetPosition(null).Y;
msg += "Y = " + y + "\n";
MessageBox.Show(msg, "SimpleEvent", MessageBoxButton.OK);
}

6. Build and run. Click over the second rectangle. The first message box will show coordinates relative to the rectangle:

The second message box will show coordinates relative to the Silverlight control: