FIREBRAND

Microsoft

MCSD: Windows Store Style Apps Using C# Certification

70-485: Advanced Store style App Development using C#

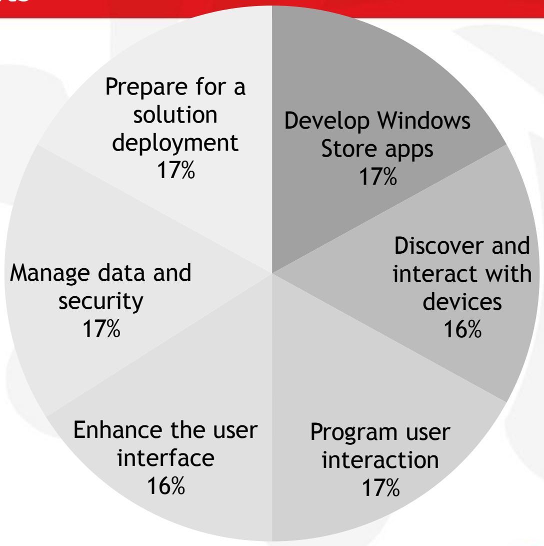
Version 1.0

www.firebrandtraining.com

Module 1
Windows Store App Essentials

Course and Exam Contents

- 55 questions
- 130 minutes
- 3 case studies



MSDN study links for Exam 70-485: Advanced Metro style App Development using C#
http://www.jayway.com/2012/12/05/msdn-study-links-for-exam-70-485-advanced-metro-style-app-development-using-c/

1.3

Common Questions

async and await work as a pair

- By using the new async and await keywords, you can use resources to create an asynchronous method almost as easily as you create a synchronous method

async modifier, Task<T> return type, Async suffix for name

```
private async Task<int> AccessTheWebAsync()
{
  HttpClient client = new HttpClient();
  Task<string> getStringTask =
    client.GetStringAsync("http://msdn.microsoft.com");
  DoIndependentWork(); // executes while async op works
  string urlContents = await getStringTask;
  return urlContents.Length;
}
```

Waits until task is complete, control returns to the caller of AccessTheWebAsync

1.4

Common Questions

Catching Exceptions with Asynchronous Operations

- As long as you call the asynchronous operation inside a try block, any exception will get caught

```
private async Task<int> AccessTheWebAsync() {
  HttpClient client = new HttpClient();
  try {
    Task<string> getStringTask =
      client.GetStringAsync("http://msdn.microsoft.com");
    DoIndependentWork(); // executes while async op works
    string urlContents = await getStringTask;
    return urlContents.Length;
  }
  catch (Exception ex) {
    // handle exception
  }
}
```

Asynchronous Tasks

- To execute any method as an asynchronous task, use a generic task factory
  - Calling StartNew is functionally equivalent to creating a Task<TResult> using one of its constructors and then calling Start to schedule it for execution

```
var t = Task<string>.Factory.StartNew(() => GetName());
string name = await t;
```

TaskFactory.StartNew<TResult> Method (Func<TResult>)
http://msdn.microsoft.com/en-us/library/dd321455.aspx

Module 2
Implementing Animations and Transitions

Implementing Animations and Transitions

Contents

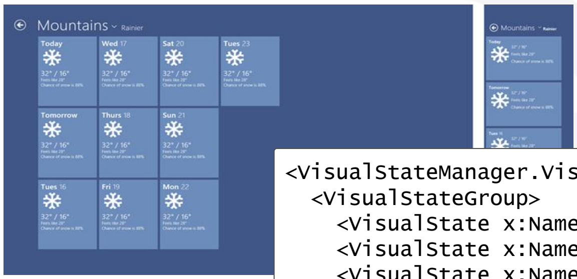
Exam Topic: Create animations and transitions
- ❑ Apply animations from the animation library
- ❑ Create and customize animations and transitions, including XAML transitions
- ❑ Implement storyboards and transformations
- ❑ Utilize built-in animations for controls

Animating your UI (Windows Store apps using C#/VB/C++ and XAML)
`http://msdn.microsoft.com/library/windows/apps/hh452701.aspx`

2.3

Visual States

- Apps run on a variety of screen sizes and under various view states
  - A user might have your app snapped to the side of a 25-inch desktop monitor, or fill the screen of a 10-inch tablet



```
<VisualStateManager.VisualStateGroups>
  <VisualStateGroup>
    <VisualState x:Name="FullScreenLandscape"/>
    <VisualState x:Name="Filled"/>
    <VisualState x:Name="FullScreenPortrait"> ...
    <VisualState x:Name="Snapped"> ...
```

Getting the most out of your pixels - adapting to view state changes
http://blogs.msdn.com/b/windowsappdev/archive/2012/04/19/getting-the-most-out-of-your-pixels-adapting-to-view-state-changes.aspx

2.4

Animation

Property Path Syntax

- For animations, a property path is used to define the connection between the named animation target object's property and the intended target animation property, by traversing object-property relationships in the property values

```
<TransformGroup x:Key="MyTransformGroupResource">
  <ScaleTransform />
  <RotateTransform />
```

```
<Rectangle Name="Rectangle02"
  RenderTransform="{StaticResource MyTransformGroupResource}">
```

```
<DoubleAnimation Storyboard.TargetName="Rectangle02"
  Storyboard.TargetProperty="RenderTransform.Children[1].Angle" ... />
```

PropertyPath XAML Syntax
http://msdn.microsoft.com/en-us/library/ms742451.aspx

Storyboards Overview
http://msdn.microsoft.com/en-us/library/ms742868.aspx

Animation

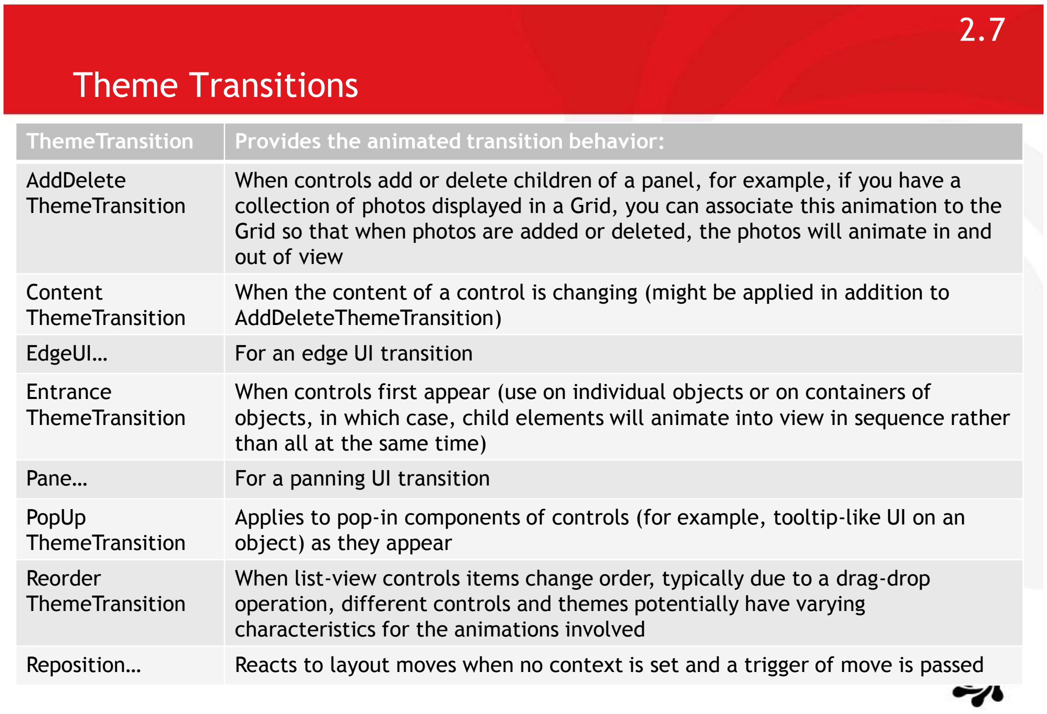
How to Create a Storyboard using Code

```
Rectangle myRectangle = new Rectangle();
// set rectangle properties with fills and so on
LayoutRoot.Children.Add(myRectangle);
Duration duration = new Duration(TimeSpan.FromSeconds(2));
DoubleAnimation myDoubleAnimation1 = new DoubleAnimation();
DoubleAnimation myDoubleAnimation2 = new DoubleAnimation();
myDoubleAnimation1.Duration = duration;
myDoubleAnimation2.Duration = duration;
Storyboard sb = new Storyboard();
sb.Duration = duration;
sb.Children.Add(myDoubleAnimation1);
sb.Children.Add(myDoubleAnimation2);
Storyboard.SetTarget(myDoubleAnimation1, myRectangle);
Storyboard.SetTarget(myDoubleAnimation2, myRectangle);
Storyboard.SetTargetProperty(
  myDoubleAnimation1, new PropertyPath("(Canvas.Left)"));
Storyboard.SetTargetProperty(
  myDoubleAnimation2, new PropertyPath("(Canvas.Top)"));
myDoubleAnimation1.To = 200;
myDoubleAnimation2.To = 200;
LayoutRoot.Resources.Add("unique_id", sb);
sb.Begin();
```

Theme Animations

ThemeAnimation	Represents the preconfigured animation that:
DragItem…	Applies to item elements being dragged
DragOver…	Applies to the elements underneath an element being dragged
DropTargetItem…	Applies to potential drop target elements
FadeIn/FadeOut ThemeAnimation	Applies to controls when they are first shown or removed from the UI or hidden
PointerUp/Down ThemeAnimation	Runs after a user taps down on an item or element (and the tap action is released)
PopIn/PopOut ThemeAnimation	Applies to pop-in components of controls (for example, tooltip-like UI on an object) as they appear/are closed/removed (this animation combines opacity and translation)
Reposition…	Use to animate an object that is being repositioned
SplitOpen/Close	Reveals a target UI using a split animation
SwipeBack/ SwipeHint	Applies to controls when an element slides back into its layout slot after a Swipe interaction or indicates that a Swipe gesture is now possible

Windows.UI.Xaml.Media.Animation classes
http://msdn.microsoft.com/en-us/library/windows/apps/jj218361.aspx



2.7

Theme Transitions

ThemeTransition	Provides the animated transition behavior:
AddDelete ThemeTransition	When controls add or delete children of a panel, for example, if you have a collection of photos displayed in a Grid, you can associate this animation to the Grid so that when photos are added or deleted, the photos will animate in and out of view
Content ThemeTransition	When the content of a control is changing (might be applied in addition to AddDeleteThemeTransition)
EdgeUI…	For an edge UI transition
Entrance ThemeTransition	When controls first appear (use on individual objects or on containers of objects, in which case, child elements will animate into view in sequence rather than all at the same time)
Pane…	For a panning UI transition
PopUp ThemeTransition	Applies to pop-in components of controls (for example, tooltip-like UI on an object) as they appear
Reorder ThemeTransition	When list-view controls items change order, typically due to a drag-drop operation, different controls and themes potentially have varying characteristics for the animations involved
Reposition…	Reacts to layout moves when no context is set and a trigger of move is passed

Module 3
Implementing Globalization and Localization

Implementing Globalization and Localization
Contents

Exam Topic: Design Windows Store apps for globalization and localization
❑ Implement .resw files to translate text
❑ Implement collation and grouping to support different reading directions
❑ Implement culture-specific formatting for dates and times

Warning! These topics sometimes appear on the 70-484 exam.

Introduction to globalization and localization
`http://channel9.msdn.com/Blogs/One-Dev-Minute/Introduction-to-globalization-and-localization`

3.3

Localization

Resource Files

- Windows 8 introduces a new resource model for Windows Store apps that replaces the hub-and-spoke model common to .NET Framework desktop apps
  - Compiled Windows Store apps use a single resource file, called a package resource index (PRI) file and stores resources for all languages, cultures, and scale factors
- To define separate languages, use sub-folders named using ISO code, each with a Resources.resw file in each
  - /fr-FR/Resources.resw, /en-GB/Resources.resw, and so on
  - The .resw file format is identical to the .resx file format, except that .resw files may contain only strings and file paths

Creating and retrieving resources in Windows Store apps
http://msdn.microsoft.com/en-us/library/windows/apps/hh694557.aspx

Quickstart: Translating UI resources (Windows Store apps using C#/VB/C++ and XAML)
http://msdn.microsoft.com/en-us/library/windows/apps/hh965329.aspx

3.4

Localization

Loading Resource Strings Using x:Uid directive

- Provides a unique identifier for markup elements
  - For Windows Runtime XAML, this unique identifier is used by XAML localization processes and tools, such as using resources from a .resw resource file

```
<Button x:Uid="GoButton" Content="Go"/>
```

- Your resource file should contain an entry for the resource named "GoButton.Content" (or just "GoButton" to set the default property for a control)
  - Content in this case is a specific property that’s inherited by the Button class
  - You might also provide localized values for other properties of this button, for example you could provide a resource-based value for "GoButton.FlowDirection"

x:Uid directive
http://msdn.microsoft.com/en-us/library/windows/apps/hh758297.aspx

Localization

Loading Resource Strings Using Code

- ResourceLoader class provides simplified access to app resources such as app UI strings
  - GetString: Returns the most appropriate string value of a resource, specified by resource identifier

```
var loader = new Windows.ApplicationModel.Resources.ResourceLoader();
var text = loader.GetString("Farewell");
```

  - GetStringForReference: Returns the most appropriate string value of a resource, specified as a Uri for a resource identifier
  - GetStringForUri: Returns the most appropriate string value of a resource, specified by a Uniform Resource Identifier (URI) resource identifier

ResourceLoader class
http://msdn.microsoft.com/en-us/library/windows/apps/windows.applicationmodel.resources.resourceloader.aspx

Module 4
Branding and a Seamless UI

Branding and a Seamless UI
Contents

No Exam Topics!

5.1

Module 5
Advanced Data Scenarios in a Windows Store App

5.2

Advanced Data Scenarios in a Windows Store App

Contents

Exam Topic: Save and retrieve files from the file system
- ❑ Handle file streams
- ❑ Save and retrieve files by using StorageFile and StorageFolder classes
- ❑ Set file extensions and associations
- ❑ Save and retrieve files by using the file picker classes
- ❑ Compress files to save space
- ❑ Access libraries, including pictures, documents, and videos

Exam Topic: Design and implement data caching
- ❑ Choose which types of items (user data, settings, app data) in an app should be persisted to the cache according to requirements
- ❑ Choose when items are cached
- ❑ Choose where items are cached (Windows Azure, remote storage)
- ❑ Select the caching mechanism

Accessing data and files (Windows Store apps using C#/VB/C++ and XAML)
`http://msdn.microsoft.com/en-us/library/windows/apps/hh758319.aspx`

Optimize loading XAML (Windows Store apps using C#/VB/C++ and XAML)
`http://msdn.microsoft.com/library/windows/apps/hh994641.aspx`

5.3

File Access Declarations

- Apps that need programmatic access to user resources such as the Documents library or removable storage must declare the appropriate capability
  - The documentsLibrary capability provides programmatic access to the user's Documents library, filtered to the file type associations declared in the package manifest, to support offline access to SkyDrive
  - The removableStorage capability provides programmatic access to files on removable storage, such as USB keys and external hard drives, filtered to the file type associations declared in the package manifest
  - For example, if a DOC reader app declared a .doc file type association, it can open .doc files in the Documents library, but not other types of files

App capability declarations (Windows Store apps)
`http://msdn.microsoft.com/en-us/library/windows/apps/hh464936.aspx`

5.4

Access the File System Efficiently

- Accessing files can be expensive due to disk latency and memory/CPU cycles to store the data
  - When you want to access a large collection of files and you want to access property values other than the typical Name, FileType, and Path properties, access them by creating QueryOptions and calling SetPropertyPrefetch

```
var queryOptions = new Windows.Storage.Search
  .QueryOptions(CommonFileQuery.OrderByDate, null);
queryOptions.SetThumbnailPrefetch(ThumbnailMode.PicturesView,
  100, ThumbnailOptions.ReturnOnlyIfCached);
queryOptions.SetPropertyPrefetch(
  PropertyPrefetchOptions.ImageProperties,
  new string[] {"System.Size"});
var queryResults = KnownFolders.PicturesLibrary
  .CreateFileQueryWithOptions(queryOptions);
```

Access the file system efficiently (Windows Store apps using C#/VB/C++ and XAML)
`http://msdn.microsoft.com/en-us/library/windows/apps/hh994634.aspx`

CommonFolderQuery

- Specifies whether the query is shallow or deep and the sorting criteria to use to group files into folders
  - DefaultQuery, GroupByYear, GroupByMonth, GroupByArtist, GroupByAlbum, GroupByAlbumArtist, GroupByComposer, GroupByGenre, GroupByPublishedYear, GroupByRating, GroupByTag, GroupByAuthor, GroupByType

```
var picturesLibrary = Windows.Storage.KnownFolders.PicturesLibrary;
var storageFolderQueryResults = picturesLibrary.createFolderQuery(
  Windows.Storage.Search.CommonFolderQuery.GroupByMonth);
```

- Do not confuse with read-only DateStackOption!
  - For example, if you create a QueryOptions object using CommonFolderQuery.GroupByMonth the DateStackOption property will contain the DateStackOption.Month value

CommonFolderQuery enumeration
http://msdn.microsoft.com/library/windows/apps/BR207957

FileInformationFactory

- Used to load information about files and folders from the results of a query and to bind these file system items to ListView and GridView controls
  - GetFilesAsync: Retrieves a collection of FileInformation objects that contain information about StorageFile objects
  - GetFoldersAsync: Retrieves a collection of FolderInformation objects that contain information about StorageFolder objects
  - GetItemsAsync: Retrieves a collection of IStorageItemInformation objects that contain information about all the items in the collection
  - GetVirtualizedFilesVector, GetVirtualizedFoldersVector, GetVirtualizedItemsVector: Gets a virtualized vector of IStorageItemInformation objects that can be bound to controls

FileInformationFactory class
http://msdn.microsoft.com/en-us/library/windows/apps/windows.storage.bulkaccess.fileinformationfactory

5.7

Roaming Data

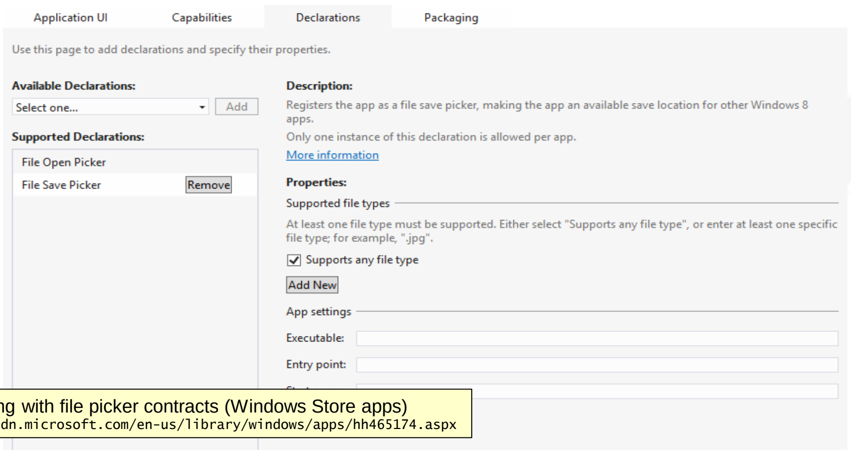
- Any user can benefit from roaming application data as long as they are using a Microsoft Account to log on to their device
- Do
  - Do use roaming for preferences and customizations
  - Do use roaming to let users continue a task across devices
- Do NOT
  - Don’t use roaming for information that is local to a device
  - Don’t use roaming to move large datasets
  - Don’t use roaming for instant syncing or for frequently changing information

Guidelines for roaming application data
`http://msdn.microsoft.com/en-us/library/windows/apps/hh465094.aspx`

5.8

Providing a Save Location

- Consider declaring the file save picker to provide your app as a location where the user can save files if your app connects the user to a service that hosts their files



Use this page to add declarations and specify their properties.

Available Declarations:
Select one... | Add

Supported Declarations:
File Open Picker
File Save Picker | Remove

Description:
Registers the app as a file save picker, making the app an available save location for other Windows 8 apps.
Only one instance of this declaration is allowed per app.
More information

Properties:
Supported file types
At least one file type must be supported. Either select "Supports any file type", or enter at least one specific file type; for example, ".jpg".
[✓] Supports any file type
Add New

App settings
Executable:
Entry point:

Integrating with file picker contracts (Windows Store apps)
`http://msdn.microsoft.com/en-us/library/windows/apps/hh465174.aspx`

Local Application Data

- Get the settings in an ApplicationDataContainer object

```
using Windows.Storage;
var localSettings = ApplicationData.Current.LocalSettings;
```

- Create a container
  - Always means the container should be created if it does not exist

```
var container = localSettings.CreateContainer("exampleContainer",
  ApplicationDataCreateDisposition.Always);
```

- Check your container exists before writing to it

```
if (localSettings.Containers.ContainsKey("exampleContainer")) {
  localSettings.Containers["exampleContainer"]
    .Values["exampleSetting"] = "Hello Windows";
```

Quickstart: Local application data (Windows Store apps using C#/VB/C++ and XAML)
`http://msdn.microsoft.com/en-us/library/windows/apps/hh700361.aspx`

6.1

Module 6
Creating Reusable Controls and Components

6.2

Creating Reusable Controls and Components
Contents

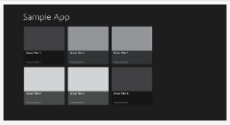
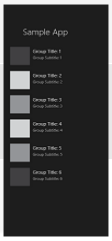
Exam Topic: Create custom controls
❑ Choose the appropriate base control to create a custom control template
❑ Style a control through control templates
❑ Design the control template to respond to changes in viewstate

Exam Topic: Create and consume WinMD components
❑ Create a WinMD component in C#
❑ Consume a WinMD component
❑ Handle WinMD reference types
❑ Reference a WinMD component

6.3

Custom Controls

Collection and Data Controls to Derive From

Control	Description	Example
FlipView	A control that presents a collection of items that the user can flip through, *one item at a time*	
GridView	A control that presents a collection of items in rows and columns that can scroll *horizontally*	
ListView	A control that presents a collection of items in a list that can scroll *vertically*	

Controls by function (Windows Store apps using C#/VB/C++ and XAML)
http://msdn.microsoft.com/en-us/library/windows/apps/hh465345.aspx

6.4

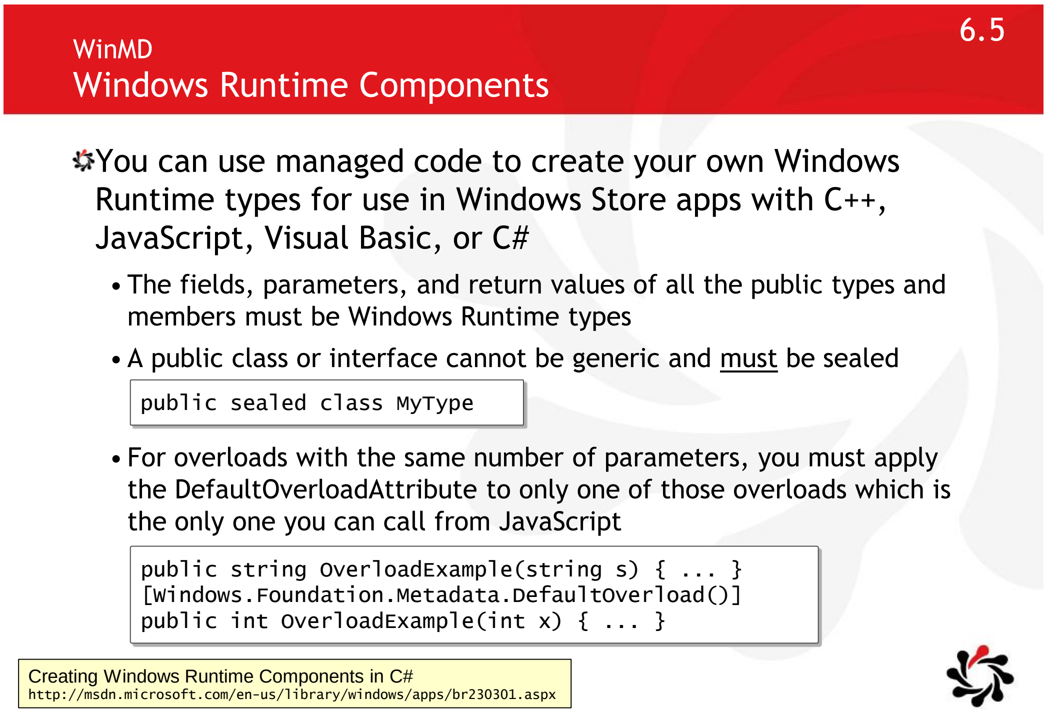
Custom Controls

Default Style Keys

- All controls have a default style that is applied automatically, for example Button, by searching for a style for the matching Type
  - In the Essentials course we saw how you can override this either using TargetType or applying a named style
- When creating custom controls, you will often want to override the default style key used otherwise your control will continue to use its parents default style

```
public class MyButton : Button {
  public MyButton() {
    this.DefaultStyleKey = typeof(MyButton);
  }
}
```

FrameworkElement.DefaultStyleKey Property
http://msdn.microsoft.com/en-us/library/system.windows.frameworkelement.defaultstylekey.aspx



WinMD

Windows Runtime Components

- You can use managed code to create your own Windows Runtime types for use in Windows Store apps with C++, JavaScript, Visual Basic, or C#
  - The fields, parameters, and return values of all the public types and members must be Windows Runtime types
  - A public class or interface cannot be generic and <u>must</u> be sealed

```
public sealed class MyType
```

  - For overloads with the same number of parameters, you must apply the DefaultOverloadAttribute to only one of those overloads which is the only one you can call from JavaScript

```
public string OverloadExample(string s) { ... }
[Windows.Foundation.Metadata.DefaultOverload()]
public int OverloadExample(int x) { ... }
```

Creating Windows Runtime Components in C#
http://msdn.microsoft.com/en-us/library/windows/apps/br230301.aspx

Module 7
Implementing Advanced Contract Scenarios

Implementing Advanced Contract Scenarios
Contents

Print from your app
`http://channel9.msdn.com/Blogs/One-Dev-Minute/Print-from-your-app`

Streaming media to devices using Play To (Windows Store apps using C#/VB/C++ and XAML)
`http://msdn.microsoft.com/library/windows/apps/hh465183.aspx`

Printing

Initialization

7.3

- The PrintManager class informs Windows that an application wishes to participate in printing
  - Also used for programmatically initiating printing
  - You must first call the GetForCurrentView method
  - You must add a listener for the PrintTaskRequested event

```
PrintManager mgr = PrintManager.GetForCurrentView();
mgr.PrintTaskRequested += mgr_PrintTaskRequested;
PrintDocument doc = new PrintDocument();
IPrintDocumentSource source = doc.DocumentSource;
// Add an event handler which creates preview pages.
doc.Paginate += CreatePrintPreviewPages;
// Add an event handler which provides a specified preview page.
doc.GetPreviewPage += GetPrintPreviewPage;
// Add an event handler which provides all final print pages.
doc.AddPages += AddPrintPages;
```

PrintManager class
http://msdn.microsoft.com/en-us/library/windows/apps/windows.graphics.printing.printmanager

Printing

PrintTaskRequested

7.4

- When a user selects a printer on the Devices charm, the PrintTaskRequested event is raised

```
protected virtual void PrintTaskRequested(
    PrintManager sender, PrintTaskRequestedEventArgs e) {
  PrintTask printTask = e.Request.CreatePrintTask(
    "C# Printing SDK Sample", sourceRequested =>
    { sourceRequested.SetSource(source); });
}
```

- After the print task is created, the PrintManager requests a collection of print pages to show in the print preview UI by raising the Paginate event

Printing (Windows Store apps using C#/VB/C++ and XAML)
http://msdn.microsoft.com/library/windows/apps/hh465196.aspx

Quickstart: Printing from your app (Windows Store apps using C#/VB/C++ and XAML)
http://msdn.microsoft.com/en-us/library/windows/apps/hh465204.aspx

Printing

PrintDocument Events

7.5

- Your app accesses Windows printing by registering for the Print contract in each view of the app from which you want users to be able to print
  - Registering for the Print contract means obtaining a PrintManager object, creating a PrintTask object, and handling the PrintDocument events

Event	Description
AddPages	Occurs when the PrintManager requests the final collection of pages to send to the printer
GetPreviewPage	Occurs when the PrintManager requests a particular print page to be shown in the preview window
Paginate	Occurs when the PrintManager requests the collection of print pages to be shown in the preview window

PlayTo

PlayToManager

7.6

- If your application includes audio, video, or image elements, users can stream the media source for those elements to a Play To target device
- The PlayToManager class has these members
  - SourceRequested event: when a user requests media to stream to a Play To target device
  - SourceSelected event: when a Play To source element has been selected
  - GetForCurrentView(): the Play To manager for the current view
  - ShowPlayToUI(): displays the Play To UI
  - DefaultSourceSelection property (default true): enables or disables the default source selection for Play To

PlayToManager class
http://msdn.microsoft.com/library/windows/apps/br206972

7.7

PlayTo

Disabling Default Source Selection

- An app that contains media elements has Play To enabled by default
  - If a user invokes the Devices charm while running the app and selects a target device to stream media to, Play To will stream the media from the first audio, video, or image element on the current page
  - You can disable this default behavior by setting the DefaultSourceSelection property to false

```
var ptm = Windows.Media.PlayTo.PlayToManager.GetForCurrentView();
ptm.DefaultSourceSelection = false;
```

PlayToManager.DefaultSourceSelection
http://msdn.microsoft.com/en-US/library/windows/apps/windows.media.playto.playtomanager.defaultsourceselection

7.8

PlayTo

SourceRequested Event

- You can select which media is streamed by using the SourceRequested event
  - In Play To, video starts from the current position
  - If you want to start the video from the beginning, seek to the beginning when the Play To connection is established

```
using Windows.Media.PlayTo;
```

```
private PlayToManager ptm = PlayToManager.GetForCurrentView();
```

```
protected override void OnNavigatedTo(NavigationEventArgs e) {
    ptm.SourceRequested += sourceRequestHandler; }
```

```
private void sourceRequestHandler(PlayToManager sender,
  PlayToSourceRequestedEventArgs e) {
    e.SourceRequest.SetSource(mediaElement.PlayToSource);
}
```

Quickstart: Using Play To in applications (Windows Store apps using C#/VB/C++ and XAML)
http://msdn.microsoft.com/en-US/library/windows/apps/hh465191

PlayTo

Deferrals

- Use a deferral when you want to make an asynchronous call to retrieve the media element to stream
  - Play To will then wait for you to supply the media element until you mark the deferral as complete
  - If you create a deferral and the wait time exceeds the Deadline property, Play To will continue without a source element

```
private void sourceRequestHandler(PlayToManager sender,
    PlayToSourceRequestedEventArgs e)
{
  var deferral = e.SourceRequest.GetDeferral();
  // Async call to get source media
  var element = await getMediaElementAsync();
  e.SourceRequest.SetSource(element.PlayToSource);
  deferral.Complete();
}
```

PlayToSourceDeferral class
http://msdn.microsoft.com/en-us/library/windows/apps/windows.media.playto.playtosourcedeferral

8.1

Module 8
The Windows Push Notification Service

8.2

The Windows Push Notification Service
Contents

- The Windows Push Notification Services (WNS) enables third-party developers to send toast, tile, badge, (all UI updates) and raw (app-defined) updates from their own cloud service
  - The WNS authentication scheme is implemented using the client credentials profile from the OAuth 2.0 protocol
  - The cloud service authenticates with WNS by providing its credentials (Package SID and secret key)

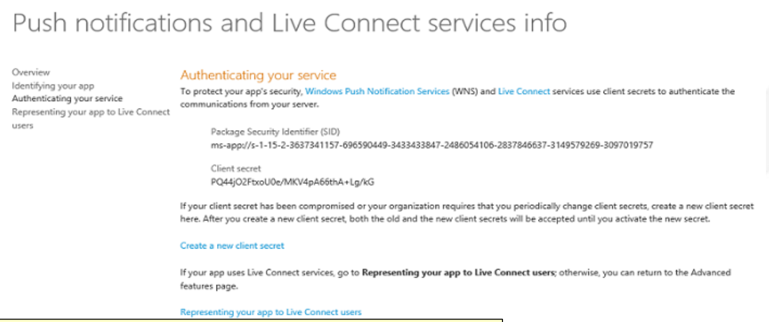
Exam Topic: Notify users by using Windows Push Notification Service (WNS)
❑ Authenticate with WNS
❑ Request, create, and save a notification channel
❑ Call and poll the WNS

Push notification overview (Windows Store apps)
http://msdn.microsoft.com/library/windows/apps/hh913756.aspx

8.3

WNS
Authentication

- Before you can send notifications through WNS, you must register your app with the Dashboard
  - When you register your app with the Dashboard, you are given credentials—a Package security identifier (SID) and a secret key—which your cloud service will use to authenticate itself



Push notifications and Live Connect services info

Overview
Identifying your app
Authenticating your service
Representing your app to Live Connect users

Authenticating your service

To protect your app's security, Windows Push Notification Services (WNS) and Live Connect services use client secrets to authenticate the communications from your server.

Package Security Identifier (SID)
ms-app://s-1-15-2-3637341157-696590449-3433433847-2486054106-2837846637-3149579269-3097019757

Client secret
PQ44jO2FtxoU0e/MKV4pA66thA+Lg/kG

If your client secret has been compromised or your organization requires that you periodically change client secrets, create a new client secret here. After you create a new client secret, both the old and the new client secrets will be accepted until you activate the new secret.

Create a new client secret

If your app uses Live Connect services, go to **Representing your app to Live Connect users**; otherwise, you can return to the Advanced features page.

Representing your app to Live Connect users

How to authenticate with the Windows Push Notification Service (WNS)
http://msdn.microsoft.com/en-us/library/windows/apps/hh465407.aspx

8.4

WNS
Send the cloud server's credentials to WNS

- Send credentials in an HTTPS authentication request, including the required parameters in the "application/x-www-for-urlencoded" format

```
POST /accesstoken.srf HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: https://login.live.com
Content-Length: 211

grant_type=client_credentials&client_id=ms-app%3a%2f%2fS-1-15-2-2972962901-
  2322836549-3722629029-1345238579-3987825745-2155616079-
  650196962&client_secret=Vex8L9WOFZuj95euaLrvSH7XyoDhLJc7&scope=notify.win
  dows.com
```

  - grant_type=client_credentials
  - client_id=[replace with Package Security Identifier (SID)]
  - client_secret=[replace with client secret]
  - scope=notify.windows.com

WNS

Authentication Response from WNS

- A response of "200 OK" indicates that the authentication was successful and that the response includes an access token for the cloud server to use with any notifications it sends, until that access token expires

```
HTTP/1.1 200 OK
Cache-Control: no-store
Content-Length: 422
Content-Type: application/json

{
    "access_token":"EgAcAQMAAAAALYAAY/c+Huwi3Fv4Ck10UrKNmtxRO6Njk2MgA=",
    "token_type":"bearer"
}
```

Push notification service request and response headers
http://msdn.microsoft.com/en-us/library/windows/apps/hh465435.aspx

WNS

Authorization

- The authorization header is used to specify the credentials of the calling party, following the OAuth 2.0 authorization method for bearer tokens
- The syntax consists of a string literal "Bearer", followed by a space, followed by your access token

```
Authorization: Bearer <access-token>
```

Authorization
http://msdn.microsoft.com/en-us/library/windows/apps/hh465435.aspx#pncodes_auth

WNS

X-WNS-Type

- These are the notification types supported by WNS. This header indicates the type of notification and how WNS should handle it
- After the notification reaches the client, the actual payload is validated against this specified type

```
X-WNS-Type: wns/toast | wns/badge | wns/tile | wns/raw
```

X-WNS-Type
http://msdn.microsoft.com/en-us/library/windows/apps/hh465435.aspx#pncodes_x_wns_type

WNS

Authentication Example (Part 1 of 2)

```
[DataContract]
public class OAuthToken
{
  [DataMember(Name = "access_token")]
  public string AccessToken { get; set; }
  [DataMember(Name = "token_type")]
  public string TokenType { get; set; }
}
```

```
private OAuthToken GetOAuthTokenFromJson(string jsonString)
{
  using (var ms = new MemoryStream(
    Encoding.Unicode.GetBytes(jsonString)))
  {
    var ser = new DataContractJsonSerializer(typeof(OAuthToken));
    var oAuthToken = (OAuthToken)ser.ReadObject(ms);
    return oAuthToken;
  }
}
```

WNS

Authentication Example (Part 2 of 2)

```
protected OAuthToken GetAccessToken(string secret, string sid)
{
  var urlEncodedSecret = HttpUtility.UrlEncode(secret);
  var urlEncodedSid = HttpUtility.UrlEncode(sid);

  var body = string.Format("grant_type=client_credentials&" +
    "client_id={0}&client_secret={1}&scope=notify.windows.com",
    urlEncodedSid, urlEncodedSecret);

  string response;
  using (var client = new WebClient())
  {
    client.Headers.Add("Content-Type",
      "application/x-www-form-urlencoded");
    response = client.UploadString(
      "https://login.live.com/accesstoken.srf", body);
  }
  return GetOAuthTokenFromJson(response);
}
```

Module 9
Capturing Media

Capturing Media
Contents

Exam Topic: Capture media with the camera and microphone
- ❑ Use CameraCaptureUI to capture pictures or video
- ❑ Use MediaCapture to capture pictures, video, or audio
- ❑ Configure camera settings
- ❑ Set media formats
- ❑ Handle media capture events

Adding multimedia (Windows Store apps using C#/VB/C++ and XAML)
`http://msdn.microsoft.com/library/windows/apps/hh465134.aspx`

9.3

Media Capture

- MediaCapture class
  - Provides functionality for capturing photos, audio, and videos from a capture device, such as a webcam
  - InitializeAsync() must be called before you can start capturing, and will launch a consent prompt to get the user's permission for the app to access the microphone or camera
- The MediaCapture class has these three properties
  - AudioDeviceController: Gets an object that controls settings for the microphone
  - MediaCaptureSettings: Gets the configuration settings for the MediaCapture object
  - VideoDeviceController: Gets an object that controls settings for the video camera

MediaCapture class
`http://msdn.microsoft.com/en-us/library/windows/apps/windows.media.capture.mediacapture.aspx`

9.4

Common Media Properties

- AudioDeviceController properties
  - Muted, VolumePercent
- MediaCaptureSettings properties
  - CameraSoundRequiredForRegion, ConcurrentRecordAndPhotoSupported, PhotoCaptureSource, StreamingCaptureMode, and so on
- VideoDeviceController properties
  - Brightness, Contrast, Exposure, FlashControl, Focus, Hue, IsoSpeedControl, Pan, Roll, Tilt, Whitebalance, Zoom, and so on
  - Properties are of type MediaDeviceControl with methods: TryGetAuto(), TryGetValue(), TrySetAuto(), TrySetValue() and a Capabilities property

MediaDeviceControl class
`http://msdn.microsoft.com/en-us/library/windows/apps/windows.media.devices.mediadevicecontrol.aspx`

9.5

Checking Media Capabilities

- Properties such as Zoom are MediaDeviceControl
  - TryGetAuto(), TryGetValue(), TrySetAuto(), TrySetValue()

```
double zoomValue = 0.0;
bool zoom = video.Zoom.TryGetValue(out zoomValue);
```

  - Capabilities property

Property	Description
AutoModeSupported	Queries whether the camera supports automatic adjustment of the setting
Default	Gets the default value of the camera setting
Max, Min	Sets the maximum or minimum value of the camera setting
Step	Gets the step size for the setting
Supported	Indicates whether the camera supports this camera setting

MediaDeviceControlCapabilities class
http://msdn.microsoft.com/en-us/library/windows/apps/windows.media.devices.mediadevicecontrolcapabilities.aspx

9.6

How to Create a File in a Known Folder

- Access to common locations that contain user content
  - In order to access the folder and libraries represented by the properties of this class, you must declare the necessary capabilities in your app manifest
  - CameraRoll, DocumentsLibrary, HomeGroup, MediaServerDevices, MusicLibrary, PIcturesLibrary, Playlists, RemovableDevices, SavedPictures, VideosLibrary
- All are of type StorageFolder class (see next slide)

KnownFolders class
http://msdn.microsoft.com/en-us/library/windows/apps/windows.storage.knownfolders

9.7

How to Create a File in a Known Folder

StorageFolder class

- CreateFileAsync(String)
- CreateFileAsync(String, CreationCollisionOption)
  - GenerateUniqueName (automatically appends a number if a file or folder already exists with that name), ReplaceExisting, FailIfExists (default), OpenIfExists

```
StorageFolder storageFolder = KnownFolders.DocumentsLibrary;
StorageFile storageFile = await storageFolder.CreateFileAsync(
  "sample.mp3", CreationCollisionOption.GenerateUniqueName);
```

StorageFolder class
http://msdn.microsoft.com/en-us/library/windows/apps/windows.storage.storagefolder.aspx

9.8

Choosing a Media Encoding Profile

Describes the encoding profile for an audio or video file

- Encoding profile includes descriptions of the audio and video encoding formats, and a description of the media container
- Methods: CreateAvi, CreateFromFileAsync, CreateFromStreamAsync, CreateM4a, CreateMp3, CreateMp4, CreateWav, CreateWma, CreateWmv
- The encoding quality Auto is a special preset that fills in the proper settings based on the current camera settings

```
using Windows.Media.MediaProperties;
```

```
var mediaProfile = MediaEncodingProfile.CreateMp3(
  AudioEncodingQuality.Auto);
```

MediaEncodingProfile class
http://msdn.microsoft.com/library/windows/apps/hh701026

Recording to a Media Encoding Profile

Some MediaCapture methods

- CapturePhotoToStorageFileAsync
- StartPreviewAsync
- StartRecordToCustomSinkAsync
- StartRecordToStorageFileAsync (most commonly used)

```
mc = new Windows.Media.Capture.MediaCapture();
await mc.StartRecordToStorageFileAsync(mediaProfile, storageFile);
```

- StartRecordToStreamAsync
- StopRecordAsync
- Media sinks are the pipeline objects that receive media data

MediaCapture.StartRecordToStorageFileAsync
http://msdn.microsoft.com/en-us/library/windows/apps/hh700863.aspx

Media Sinks
http://msdn.microsoft.com/en-us/library/windows/desktop/ms701626(v=vs.85).aspx

Module 10
Background Tasks

Background Tasks
Contents

Exam Topic: Create background tasks
- ❑ Implement the Windows.ApplicationModel.Background classes
- ❑ Implement the IBackgroundTask interface

Exam Topic: Consume background tasks
- ❑ Use timing and system triggers
- ❑ Keep communication channels open
- ❑ Request lock screen access
- ❑ Use the BackgroundTransfer class to finish downloads

Exam Topic: Design for and implement UI responsiveness
- ❑ Choose an asynchronous strategy for your app
- ❑ Implement the Task Parallel library for multi-processor utilization
- ❑ Convert asynchronous operations to tasks

Background Tasks

Links

Performance best practices for Windows Store apps using C++, C#, and Visual Basic
http://msdn.microsoft.com/library/windows/apps/hh750313.aspx

General best practices for performance
http://msdn.microsoft.com/library/windows/apps/hh994633.aspx

Asynchronous Programming with Async and Await (C# and Visual Basic)
http://msdn.microsoft.com/library/hh191443(vs.110).aspx

Quickstart: Create and register a background task (Windows Store apps using C#/VB/C++ and XAML)
http://msdn.microsoft.com/library/windows/apps/hh977055.aspx

Quickstart: Update a live tile from a background task
http://msdn.microsoft.com/library/windows/apps/jj991805.aspx

Supporting your app with background tasks (Windows Store apps using C#/VB/C++ and XAML)
http://msdn.microsoft.com/en-us/library/windows/apps/hh977056.aspx

Background Tasks

What Are They?

- Run your own lightweight code in the background by responding to triggers
  - Provide functionality when your app is suspended or not running
  - For real-time communication apps like VOIP, mail, and IM
- Background tasks implement IBackgroundTask interface
  - Has a Run method that passes in an IBackgroundTaskInstance that allows the task to interact with the current run process

```
public sealed class SampleBackgroundTask : IBackgroundTask {
  public void Run(IBackgroundTaskInstance taskInstance) {
    // do work
```

  - The implementation must reside in a WinRT component library
  - All public classes within a WinRT component are required to be sealed, so your background tasks will always be a sealed class

Background Tasks

Triggers

- Background tasks must be assigned to a single trigger that's responsible for determining when and how frequently a task will be executed

Trigger	Description
Control Channel	Used for high-availability, real-time apps that maintain open socket connections; allows the app to run in a low-power mode until data is received across the socket
Push Notification	Raw notification from the Windows Notification Service
Maintenance	Time-based, fires at a given time interval, but only if the device is plugged into a power source
Timer	Time-based, fires at a given time interval, but requires the app to be added to the lock screen to function
System Event	Register a task to a variety of predefined system events

Background Tasks in Windows Store Apps
http://visualstudiomagazine.com/articles/2013/05/01/background-tasks-in-windows-store-apps.aspx

Background Tasks

Registering with BackgroundTaskBuilder

- Registers an implementation of IBackgroundTask with a trigger, a unique name and any optional conditions
  - Each registered task must have a unique name within the scope of the app

```
var builder = new BackgroundTaskBuilder();
builder.Name = "MySampleTask";
builder.TaskEntryPoint = "BackgroundTasks.MyBackgroundTask";
builder.SetTrigger(new SystemTrigger(
  SystemTriggerType.NetworkStateChange, false));
// use builder.SetCondition(...) if you need conditions
var ret = builder.Register();
```

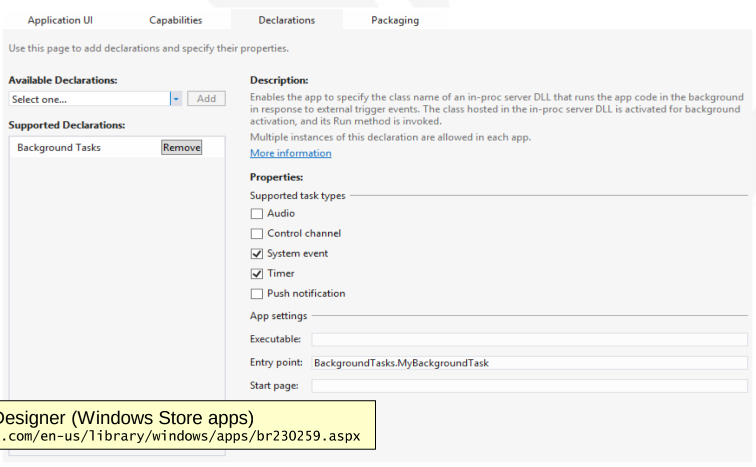
  - By default, each background task will execute within a special process, BackgroundTaskHost.exe

How to register a background task (Windows Store apps using C#/VB/C++ and XAML)
http://msdn.microsoft.com/library/windows/apps/jj553413.aspx

Background Tasks
Manifest Designer

10.7

- You can define aspects of your Windows Store app, such as background task declarations, by using the App Manifest Designer



Application UI | Capabilities | Declarations | Packaging

Use this page to add declarations and specify their properties.

Available Declarations: Select one... Add

Supported Declarations:

Background Tasks Remove

Description:

Enables the app to specify the class name of an in-proc server DLL that runs the app code in the background in response to external trigger events. The class hosted in the in-proc server DLL is activated for background activation, and its Run method is invoked.

Multiple instances of this declaration are allowed in each app.

More information

Properties:

Supported task types

- ☐ Audio
- ☐ Control channel
- ☑ System event
- ☑ Timer
- ☐ Push notification

App settings

Executable:

Entry point: BackgroundTasks.MyBackgroundTask

Start page:

Using the Manifest Designer (Windows Store apps)
`http://msdn.microsoft.com/en-us/library/windows/apps/br230259.aspx`

Background Tasks
System Events

10.8

- Respond to system-generated events by registering a background task with the SystemTrigger class

Trigger	Description
InternetAvailable	The Internet becomes available
NetworkStateChange	A network change such as a change in cost or connectivity occurs
OnlineIdConnectedStateChange	Online ID associated with the account changes
SmsReceived	A new SMS message is received by an installed mobile broadband device
TimeZoneChange	The time zone changes on the device (for example, when the system adjusts the clock for daylight saving time)

- Your app can also run tasks as frequently as every 15 minutes by using the maintenance trigger
  - Only run when the device is plugged in to AC power

Background Tasks

Conditions

- You can control when the background task runs, even after it is triggered, by adding a condition
  - Once triggered, a background task will not run until all of its conditions are met

Condition	Description
InternetAvailable	The Internet must be available
InternetNotAvailable	The Internet must be unavailable
SessionConnected	The session must be connected
SessionDisconnected	The session must be disconnected
UserNotPresent	The user must be away
UserPresent	The user must be present

Background Tasks

Lock Screen-Capable Apps

- Apps can be placed on the lock screen to show real-time information to the user at a glance
  - Real-time triggers can be used to run lightweight custom code in the background for apps that are on the lock screen

Trigger	Description
Control Channel	Background tasks can keep a connection alive, and receive messages on the control channel, by using the ControlChannelTrigger
Timer	Background tasks can run as frequently as every 15 minutes, and they can be set to run at a certain time, by using the TimeTrigger
Push Notification	Background tasks respond to the PushNotificationTrigger to receive raw push notifications

  - The user must place your app on the lock screen before the app can use these background tasks

Background Tasks

System Event Triggers for Lock Screen-Capable Apps

- The SystemTriggerType enumeration includes the following system event triggers that are only usable by lock screen-capable apps

Trigger	Description
UserPresent, UserAway	The background task is triggered when the user becomes present / absent
ControlChannelReset	The background task is triggered when a control channel is reset
SessionConnected	The background task is triggered when the session is connected
LockScreenApplicationAdded, LockScreenApplicationRemoved	An app tile is added to / removed from the lock screen

Background Tasks

Long-Running and Asynchronous Operations

- By default, once the Run method completes, the task-executing stack is terminated so any asynchronous operations won’t complete
  - Also, each task is given a short period of time to complete its execution and will be terminated if that time limit is reached
  - The IBackgroundTaskInstance which gets passed to the Run method includes a GetDeferral method that returns a BackgroundTaskDeferral instance; the executing code won’t complete until the Complete method is called

```
public void Run(IBackgroundTaskInstance taskInstance) {
  BackgroundTaskDeferral deferral = taskInstance.GetDeferral();
  // do work
  deferral.Complete();
}
```

Long-Running and Asynchronous Operations
http://visualstudiomagazine.com/Articles/2013/05/01/Background-Tasks-in-Windows-Store-Apps.aspx?Page=2

Module 11
Working with Sensors and Devices

Working with Sensors and Devices

Contents

Exam Topic: Get data from sensors
❑ Determine the availability of a sensor (Windows.Devices.Sensors)
❑ Add sensor requests to the app manifest
❑ Handle sensor events
❑ Get sensor properties
❑ Determine location via GPS

Exam Topic: Enumerate and discover device capabilities
❑ Discover the capabilities of a device (for example, GPS, accelerometer, near field communication, and camera)

Guidelines for location-aware apps (Windows Store apps using C#/VB/C++ and XAML)
`http://msdn.microsoft.com/en-us/library/windows/apps/hh465127.aspx`

Responding to motion and orientation sensors (Windows Store apps using C#/VB/C++ and XAML)
`http://msdn.microsoft.com/library/windows/apps/hh465294.aspx`

Responding to light sensors
`http://msdn.microsoft.com/library/windows/apps/hh465287.aspx`

Sensors
Types

11.3

Sensor	Description
Accelerometer	Detects acceleration along three axes (x, y, and z)
Inclinometer	Detects angle of incline along three axes (pitch, roll, and yaw)
Gyrometer	Detects angular velocity along three axes
Compass	Detects heading in degrees relative to magnetic north (and due north when integrated with onboard GPS)
Light	Detects ambient lighting level in lumens
Orientation	Combines data from the accelerometer, compass, and gyrometer sensors to provide smoother and more sensitive rotation data than can be obtained from any of the sensors alone
Simple Orientation	Uses the accelerometer to obtain device orientation as a rotation into one of four quadrants, or face-up, or face-down

Windows.Devices.Sensors namespace
http://msdn.microsoft.com/en-us/library/windows/apps/windows.devices.sensors.aspx

Sensors
Detecting Orientation

11.4

- This sensor returns a rotation matrix and a Quaternion that can be used to adjust the user's perspective

```
if(OrientationSensor.GetDefault() != null) // if sensor exists
  _sensor = OrientationSensor.GetDefault();
```

```
-sensor.ReportInterval = 15000; // 15 seconds minimum interval
_sensor.ReadingChanged += new TypedEventHandler<OrientationSensor,
  OrientationSensorReadingChangedEventArgs>(ReadingChanged);
```

```
async private void ReadingChanged(object sender,
                OrientationSensorReadingChangedEventArgs e) {
  await Dispatcher.RunAsync(CoreDispatcherPriority.Normal, () =>
    {
      SensorQuaternion quaternion = e.Reading.Quaternion;
      // quaternion.X, quaternion.Y, quaternion.Z, quaternion.W
      SensorRotationMatrix rm = e.Reading.RotationMatrix;
      // rm.M11, M12, M13, M21, M22, M23, M31, M32, M33
```

OrientationSensor class
http://msdn.microsoft.com/en-us/library/windows/apps/windows.devices.sensors.orientationsensor.aspx

Determine location

Detecting Geolocation

11.5

- Subscribe to location updates so that your app can respond to location changes

```
private Geolocator geo = new Geolocator();
```

```
geo.ReportInterval = 15000; // 15 seconds minimum interval
geo.PositionChanged += new TypedEventHandler
  <Geolocator, PositionChangedEventArgs>(geo_PositionChanged);
```

```
private void geo_PositionChanged(Geolocator sender,
                                 PositionChangedEventArgs e) {
  IGeoposition pos =
    (e.Context as IPositionChangedEventArgs).Position;
  // pos.Coordinate.Latitude.ToString();
  // pos.Coordinate.Longitude.ToString();
  // pos.Coordinate.Accuracy.ToString();
}
```

How to respond to location updates (Windows Store apps using C#/VB/C++ and XAML)
http://msdn.microsoft.com/en-us/library/windows/apps/hh465142.aspx

Determine location

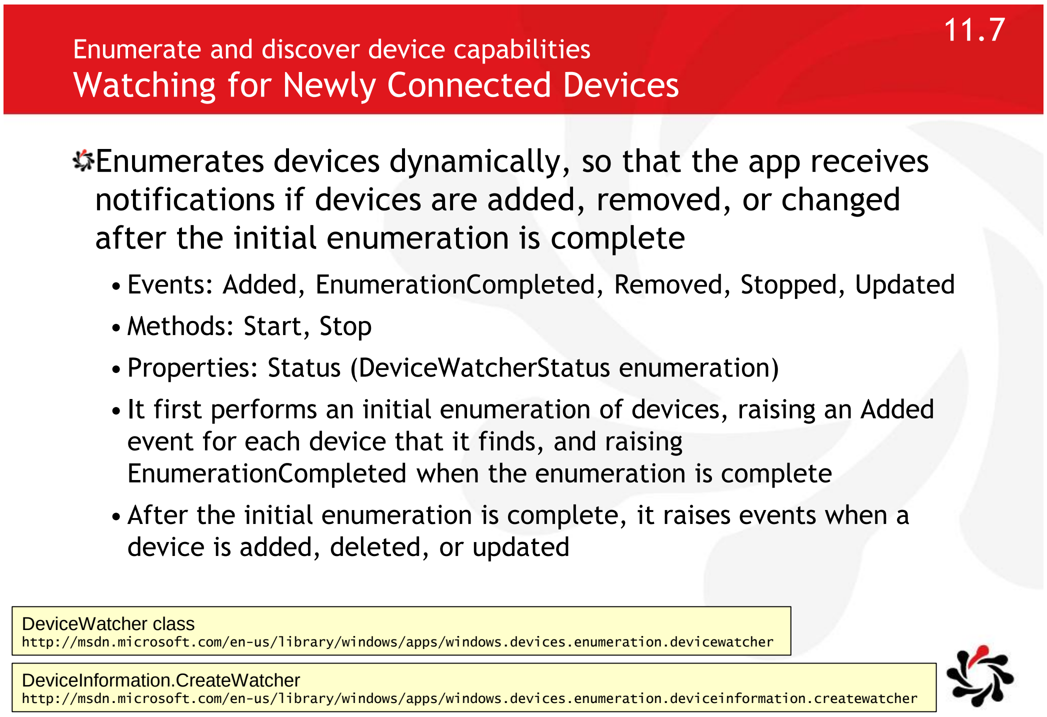
Checking the Status of Location Tracking

11.6

- PositionStatus indicates the ability of the Geolocator object to provide location data

Member	Description
Ready	Location data is available
Initializing	This is the status if a GPS is the source of location data and the GPS receiver does not yet have the required number of satellites in view to obtain an accurate position
NoData	No location data is available from any location provider
Disabled	Indicates that the user has not granted the application permission to access location
NotInitialized	If the application has not yet called GetGeopositionAsync or registered an event handler for the PositionChanged event
NotAvailable	The Windows Sensor and Location Platform is not available on this version of Windows

PositionStatus enumeration
http://msdn.microsoft.com/en-us/library/windows/apps/windows.devices.geolocation.positionstatus



Enumerate and discover device capabilities

Watching for Newly Connected Devices

- Enumerates devices dynamically, so that the app receives notifications if devices are added, removed, or changed after the initial enumeration is complete
  - Events: Added, EnumerationCompleted, Removed, Stopped, Updated
  - Methods: Start, Stop
  - Properties: Status (DeviceWatcherStatus enumeration)
  - It first performs an initial enumeration of devices, raising an Added event for each device that it finds, and raising EnumerationCompleted when the enumeration is complete
  - After the initial enumeration is complete, it raises events when a device is added, deleted, or updated

DeviceWatcher class
http://msdn.microsoft.com/en-us/library/windows/apps/windows.devices.enumeration.devicewatcher

DeviceInformation.CreateWatcher
http://msdn.microsoft.com/en-us/library/windows/apps/windows.devices.enumeration.deviceinformation.createwatcher

Module 12
Generating Revenue with Your Windows Store App

Generating Revenue with Your Windows Store App

Contents

Exam Topic: Design and implement trial functionality in an app
- ❑ Set up a timed trial
- ❑ Set up a feature-based trial
- ❑ Set up in-app purchases
- ❑ Transition an app from trial to full

Certify your app
`http://msdn.microsoft.com/library/windows/apps/hh694079.aspx`

CurrentAppSimulator

- Defines methods and properties used to instantiate an object that you can use to get simulated license info during testing
  - Until the app has been listed in the Windows Store, the CurrentApp object won't work in the app
  - Use the CurrentAppSimulator to test your app’s licensing and in-app purchases while you develop your app
  - After you test your app and before you submit it to the Windows Store, replace CurrentAppSimulator with CurrentApp
  - The CurrentAppSimulator object gets its data from the WindowsStoreProxy.xml file (example on next slide) in the %userprofile%\appdata\local\packages\<package-moniker>\localstate\microsoft\Windows Store\Apidata folder

CurrentAppSimulator class
http://msdn.microsoft.com/en-us/library/windows/apps/windows.applicationmodel.store.currentappsimulator

WindowsStoreProxy.xml

```
<?xml version="1.0" encoding="UTF-16"?>
<CurrentApp>
  <ListingInformation>
    <App>
      <AppId>2B14D306-D8F8-4066-A45B-0FB3464C67F2</AppId>
      <LinkUri>http://apps.windows.microsoft.com/app/2B14D306-D8F8-4066-A45B-0FB3464C67F2</LinkUri>
      <CurrentMarket>en-US</CurrentMarket>
      <AgeRating>3</AgeRating>
      <MarketData xml:lang="en-us">
        <Name>Trial management full license</Name>
        <Description>Sample app for demonstrating trial license management</Description>
        <Price>4.99</Price>
        <CurrencySymbol>$</CurrencySymbol>
      </MarketData>
    </App>
  </ListingInformation>
  <LicenseInformation>
    <App>
      <IsActive>true</IsActive>
      <IsTrial>true</IsTrial> <!-- set to true to test how app behaves in trial mode-->
      <ExpirationDate>2012-01-19T05:00:00.00Z</ExpirationDate>
    </App>
  </LicenseInformation>
  <Simulation SimulationMode="Automatic">
    <DefaultResponse MethodName="LoadListingInformationAsync_GetResult" HResult="E_FAIL"/>
  </Simulation>
</CurrentApp>
```

12.5

Trial Apps and Features

- If customers can use your app for free during a trial period, you can design your app to exclude or limit some features during the trial period
  - The current license state of your app is stored as properties of the LicenseInformation class
  - If the customer buys your app while it is running, your app can silently enable the features that are available with a full-license (or disable the trial-only notices)
  - Initialize the CurrentAppSimulator to access the app's license

```
licenseInformation = CurrentAppSimulator.LicenseInformation;
```

  - **Warning!** Before you submit your app to the Windows Store for certification, replace replace all uses of CurrentAppSimulator with CurrentApp

Create a trial version of your app
http://msdn.microsoft.com/library/windows/apps/hh694065.aspx

12.6

How to Enable In-App Purchases

- For each in-app purchase feature, create an in-app offer and add it to your app

```
if (!licenseInformation.ProductLicenses["featureName"].IsActive) {
  try
  {
    // Customer doesn't own this feature, so show purchase dialog.
    await CurrentAppSimulator.RequestProductPurchaseAsync(
      "featureName", false);
    // Check the license state to determine if the in-app purchase
    // was successful.
  }
  catch (Exception) {
    // The in-app purchase was not completed; an error occurred.
  }
} else {
  // The customer already owns this feature.
}
```

Enable in-app purchases from your app
http://msdn.microsoft.com/en-us/library/windows/apps/hh694067.aspx

Module 13
Securing Windows Store App Data

Securing Windows Store App Data

Contents

OAuth2 Authentication

- Supported OAuth flows
  - Implicit grant flow
  - Authorization code grant flow
  - Sign-in control flow

Access Online Services with the Windows Runtime and OAuth
http://msdn.microsoft.com/en-us/magazine/jj883954.aspx

OAuth 2.0
http://msdn.microsoft.com/en-us/library/live/hh243647.aspx

OAuth 2.0
http://oauth.net/2/

Encrypting Data

- You can protect data to the
  - Local user or computer account
    - "LOCAL=user" or "LOCAL=machine"
  - Credentials (password) used during logon to a website
    - "WEBCREDENTIALS=MyPasswordName,myweb.com"

```
string strMsg = "This is a message to be protected.";
string strDescriptor = "LOCAL=user";
var encoding = BinaryStringEncoding.Utf8;
var provider = new DataProtectionProvider(strDescriptor);
IBuffer buffMsg = CryptographicBuffer
  .ConvertStringToBinary(strMsg, encoding);
IBuffer buffProtected = await provider.ProtectAsync(buffMsg);
```

DataProtectionProvider class
http://msdn.microsoft.com/en-us/library/windows/apps/windows.security.cryptography.dataprotection.dataprotectionprovider

14.1

Module 14
Tracing and Profiling Windows Store Apps

14.2

Tracing and Profiling Windows Store Apps
Contents

Exam Topic: Design for error handling
- ❑ Design the app so that errors and exceptions never reach the user
- ❑ Application class for global collection
- ❑ Handle device capability errors
- ❑ Handle asynchronous errors

Exam Topic: Design and implement a test strategy
- ❑ Recommend a functional test plan
- ❑ Implement a coded UI test
- ❑ Recommend a reliability test plan (performance testing, stress testing, scalability testing, duration testing)
- ❑ Implement unit testing in an app

Exam Topic: Design a diagnostics and monitoring strategy
- ❑ Design profiling, tracing, performance counters, audit trails (events and information), and usage reporting
- ❑ Decide where to log events (local vs. centralized reporting)

Debugging and testing with Visual Studio
`http://msdn.microsoft.com/library/windows/apps/hh441481.aspx`

14.3

Error Handling

- Normally after the UnhandledException event is raised, the XAML framework terminates the application because the exception was unhandled
  - The application has some control over this – if the UnhandledException event handler sets the Handled property of the event arguments to true, then in most cases the application will not be terminated

```
public class App : Application
{
    public App()
    {
        UnhandledException += (sender, e) => e.Handled = true;
    }
}
```

Application.UnhandledException event
http://msdn.microsoft.com/en-us/library/windows/apps/windows.ui.xaml.application.unhandledexception

14.4

Environment for Performance Testing

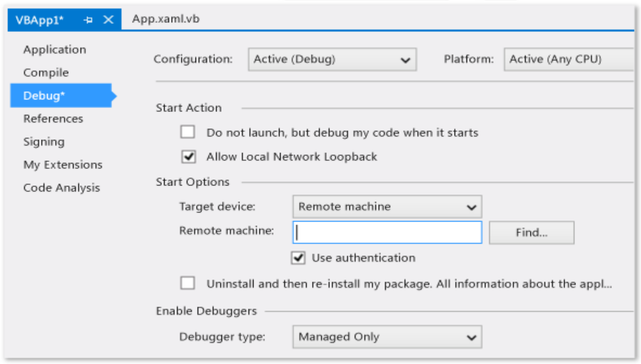
- When testing performance you must use
  - Release build
  - Local Machine deployment target or remote physical device
- Do NOT use
  - Debug build
  - Simulator deployment target

How to profile Visual C# code in Windows Store apps on a local machine
http://msdn.microsoft.com/en-us/library/hh696631.aspx

14.5

Environment for Hardware Device Testing

- When testing your app on a hardware device, set
  - Target device: Remote Machine
  - Remote machine: <name of device>



Configuration: Active (Debug)
Platform: Active (Any CPU)

Start Action
- [ ] Do not launch, but debug my code when it starts
- [x] Allow Local Network Loopback

Start Options
Target device: Remote machine
Remote machine:
Find...
- [x] Use authentication
- [ ] Uninstall and then re-install my package. All information about the appl...

Enable Debuggers
Debugger type: Managed Only

Start a debugging session in for a Windows Store app Visual Studio (Visual C++, Visual C#, and Visual Basic)
`http://msdn.microsoft.com/en-us/library/windows/apps/hh781607(v=vs.120).aspx`

14.6

Choose the Debugger Type to Use

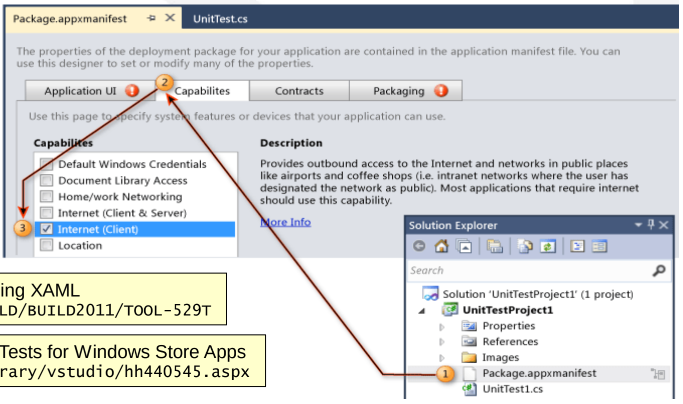
- By default, Visual Studio debugs managed code in C# and Visual Basic apps
  - Select the Enable unmanaged code debugging check box to include native code in your debug session
- By default, Visual Studio debugs native code in C++ apps

Debugger	Description
Script Only	Debug JavaScript code in your app, other code is ignored
Native Only	Debug native C/C++ code in your app, other code is ignored
Managed Only	Debug managed code in your app, other code is ignored
Mixed	Debug native C/C++ code and managed code in your app
GPU only	Debug native C++ code that runs on a graphics processing unit (GPU) on a discrete graphics card

14.7

Setting Up Unit Tests

- To set up unit tests
  - Create unit test projects (add to your app solution)
  - Edit the Manifest for the Unit Test Project (Package.appxmanifest)
  - Code and Run the Unit Test



The properties of the deployment package for your application are contained in the application manifest file. You can use this designer to set or modify many of the properties.

Use this page to specify system features or devices that your application can use.

- [ ] Default Windows Credentials
- [ ] Document Library Access
- [ ] Home/work Networking
- [ ] Internet (Client & Server)
- [x] Internet (Client)
- [ ] Location

Provides outbound access to the Internet and networks in public places like airports and coffee shops (i.e. intranet networks where the user has designated the network as public). Most applications that require internet should use this capability.

Solution 'UnitTestProject1' (1 project)
UnitTestProject1
Properties
References
Images
Package.appxmanifest
UnitTest1.cs

Unit testing your metro style apps built using XAML
`http://channel9.msdn.com/Events/BUILD/BUILD2011/TOOL-529T`

Walkthrough: Creating and Running Unit Tests for Windows Store Apps
`http://msdn.microsoft.com/en-us/library/vstudio/hh440545.aspx`

14.8

Monitoring Your Apps

- Analytics and telemetry are two types of data that Microsoft collects to help you monitor your apps in the Windows Store and after they have been installed on your customers' computers
  - **Analytics** refers to the data we collect directly from the Windows Store, such as app listing views, downloads, and customer ratings and reviews
  - **Telemetry** refers to the data we collect about your app when it's running on customers' computers. If you enable this feature in your Windows Store developer account, your app will automatically send info back to Microsoft about how often it has been launched, how long it has been running, and whether it has experienced an error such as crashing or encountering a JavaScript exception

Collecting telemetry data from your apps
`http://msdn.microsoft.com/en-us/library/windows/apps/hh967787.aspx`