**Hướng dẫn thực hành Lab**

*Music Player*

Lab version: 1.0.0

Last updated: 3/28/2012

Mục lục

[Tổng quan 3](#_Toc320744711)

[Mục tiêu 3](#_Toc320744712)

[Yêu cầu 3](#_Toc320744713)

[Ước lượng thời gian hoàn thành 3](#_Toc320744714)

[Thực hành 4](#_Toc320744715)

[Task 1 – Thêm giao diện người dùng chính và chức năng 4](#_Toc320744716)

[Task 2 – Thêm các nút điều khiển 10](#_Toc320744717)

[Task 3 – Copy các file nhạc và file ảnh vào Isolated Storage. 16](#_Toc320744718)

[Task 4 – Thêm và cài đặt Audio Playback Agent. 18](#_Toc320744719)

[Tổng kết 24](#_Toc320744720)

# **Tổng quan**

* 1. Trong phiên bản gốc đầu tiên của hệ điều hành Windows® Phone, chúng ta có thể tạo một trình phát nhạc nhưng ứng dụng này sẽ không thể tiếp tục phát nhạc khi đang chạy nền.
  2. Như chúng ta thấy trong bài lab này, với Windows® Phone Codenamed Mango, một ứng dụng sẽ có thể tiếp tục phát nhạc ngay cả khi đang chạy nền.. Để làm được điều này, chúng ta sẽ sử dụng lớp **Background Audio Player** mới trong namespace **Microsoft.Phone.BackgroundAudio.**
  3. Bài lab này sẽ hướng dẫn cách sử dụng nhạc nên trong ứng dụng của bạn bằng cách sử dụng Windows® Phone Codenamed Mango API, phát triển trình phát nhạc có khả năng tiếp tục phát nhạc ngay cả khi đang chạy nền.

## Mục tiêu

* 1. Bài lab này sẽ giúp bạn :
  + Sử dụng lớp BackgroundAudio để phát nhạc ngay cả khi ứng dụng của bạn đang chạy nền.
  + Cài đặt một background agent vào trong ứng dụng của bạn. ( Giúp cho ứng dụng của bạn có thể chạy ngầm dưới hệ thống và thông báo với người dùng khi hoàn thành một nhiệm vụ hay mỗi khi xảy ra một sự kiện nào đó)

## Yêu cầu

* + Microsoft Visual Studio 2010 hoặc Microsoft Visual C# Express 2010, và Windows® Phone Developer Tools download tại <http://go.microsoft.com/?linkid=9772716>
  + Có kiến thức về phát triển ứng dụng trên Windows® Phone (Nếu đây là lần tiên bạn phát triển ứng dụng trên Windows® Phone 7, hãy tham khảo tại <http://msdn.microsoft.com/en-us/gg266499>. Bài lab này sẽ xem như là bạn đã hiểu cơ bản và tập trung vào trình bày phần âm thanh nền.)

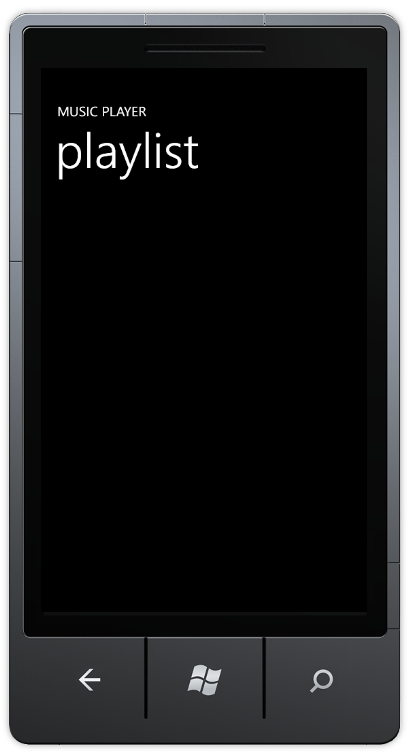
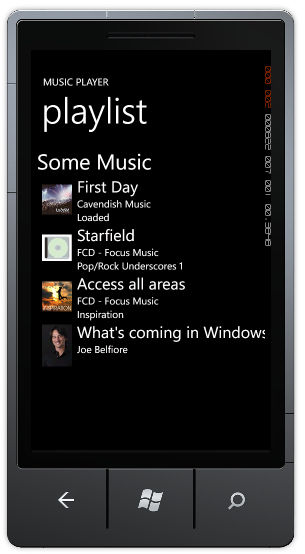
## Ước lượng thời gian hoàn thành

* 1. Bạn sẽ mất khoảng 20-40 phút để hoàn thành bài lab này.

# Thực hành

* 1. Bắt đầu bài lab với solution đã được khởi tạo sẵn trong thư mục **Source\Begin.**
  2. Solution này chứa một project với tất cả giao diện người dùng (user interface viết tắt là UI) mà chúng ta sẽ sử dụng trong bài lab này và 2 model (**Playlist** và **Track** ) mà chúng ta sẽ làm việc với chúng.

## Task 1 – Thêm giao diện người dùng chính và chức năng

1. Trong task này, chúng ta sẽ làm việc trên phần chính của ứng dụng, cài đặt các xử lý sự kiện cho **Microsoft.Phone.BackgroundAudio.BackgroundAudioPlayer** để có thể tiếp tục phát âm thanh ngay cả khi ứng dụng đang chạy nền.
   1. Mở solution trong thư mục **Source\Begin** chạy project.
   2. Mở file **MainPage.xaml.**
   3. Đặt thuộc tính **DataContext**  của chính trang này bằng cách thêm thuộc tính sau đây vào phần tử **phone:PhoneApplicationPage**:
      1. XAML
      2. DataContext="{Binding RelativeSource={RelativeSource Self}}"
   4. Xác định grid có tên **ContentPanel** và thêm thuộc tính **HorizontalAlignment** có giá trị **Stretch** cho grid này.
      1. XAML
      2. <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0" HorizontalAlignment="Stretch">
   5. Bây giờ thêm vào **RowDefinitions**  và tạo mới một **TextBlock** như sau. Bạn có thể thấy rằng chúng ta sử dụng một TextBlock mới để hiển thị tên của playlist đang hoạt động. Chúng ta sẽ thêm vào một thuộc tính để cố định playlist đang hoạt động sau đó trong bài lab.
      1. XAML
      2. <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0" HorizontalAlignment="Stretch">
      3. <Grid.RowDefinitions>
      4. <RowDefinition Height="Auto"/>
      5. <RowDefinition Height="\*"/>
      6. <RowDefinition Height="Auto"/>
      7. </Grid.RowDefinitions>
      8. <TextBlock Text="{Binding ActivePlaylist.Name}" FontSize="{StaticResource PhoneFontSizeExtraLarge}"/>
      9. </Grid>
   6. Thêm mới một **ListBox**  giống như đoạn code bên dưới. Listbox này sẽ hiển thị danh sách các track nhạc có thể sử dụng trong ứng dụng.
      1. XAML
      2. <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0" HorizontalAlignment="Stretch">
      3. <Grid.RowDefinitions>
      4. <RowDefinition Height="Auto"/>
      5. <RowDefinition Height="\*"/>
      6. <RowDefinition Height="Auto"/>
      7. </Grid.RowDefinitions>
      9. <TextBlock Text="{Binding ActivePlaylist.Name}" FontSize="{StaticResource PhoneFontSizeExtraLarge}"/>
      10. <ListBox ItemsSource="{Binding ActivePlaylist.Tracks}" Grid.Row="1" x:Name="lstTracks" HorizontalContentAlignment="Stretch" Loaded="lstTracks\_Loaded">
      11. <ListBox.ItemTemplate>
      12. <DataTemplate>
      13. <Grid HorizontalAlignment="Stretch">
      14. <Grid.RowDefinitions>
      15. <RowDefinition/>
      16. <RowDefinition/>
      17. <RowDefinition/>
      18. </Grid.RowDefinitions>
      19. <Grid.ColumnDefinitions>
      20. <ColumnDefinition Width="Auto"/>
      21. <ColumnDefinition Width="\*"/>
      22. </Grid.ColumnDefinitions>
      24. <Image Margin="10,0" Source="{Binding Tile, TargetNullValue={StaticResource NoArt}}" Grid.RowSpan="3" Width="60" Stretch="Uniform" VerticalAlignment="Center"/>
      25. <TextBlock Text="{Binding Title}" Grid.Column="1" FontSize="{StaticResource PhoneFontSizeLarge}"/>
      26. <TextBlock Text="{Binding Artist}" Grid.Row="1" Grid.Column="1"/>
      27. <TextBlock Text="{Binding Album}" Grid.Row="2" Grid.Column="1"/>
      28. </Grid>
      29. </DataTemplate>
      30. </ListBox.ItemTemplate>
      31. </ListBox>
      32. </Grid>
   7. Mở file có tên **MainPage.xaml.cs**.
   8. Thêm một phương thức rỗng có tên **lstTracks\_Loaded** như bên dưới:
      1. C#
      2. public partial class MainPage : PhoneApplicationPage
      3. {
      4. ...
      5. private void lstTracks\_Loaded(object sender, RoutedEventArgs e)
      6. {
      7. }
      8. ...
      9. }
   9. Biên dịch và chạy ứng dụng lên máy ảo, bạn sẽ được như hình bên dưới:
      * 1. 
        2. Hình 1
        3. Ứng dụng hiện tại không có playlist.
   10. Thêm một thuộc tính cho playlist đang hoạt động mà chúng ta dùng để liên kết trong bước vừa rồi. Thêm một **Playlist** DependencyProperty giống như bên dưới và gọi **ActivePlaylist.**
       1. C#
       2. public partial class MainPage : PhoneApplicationPage
       3. {
       4. public Playlist ActivePlaylist
       5. {
       6. get { return (Playlist)GetValue(ActivePlaylistProperty); }
       7. set { SetValue(ActivePlaylistProperty, value); }
       8. }
       9. public static readonly DependencyProperty ActivePlaylistProperty =
       10. DependencyProperty.Register("ActivePlaylist", typeof(Playlist), typeof(MainPage), new PropertyMetadata(null));
       11. ...
       12. }
   11. Override phương thức **OnNavigatedTo**  để triển khai playlist được cung cấp như là một trong những resoucre của ứng dụng vào isolated storage. (*isolated storage : vùng nhớ để các ứng dụng lưu trữ các thông tin về cấu hình…*)
       1. C#
       2. public partial class MainPage : PhoneApplicationPage
       3. {
       4. ...
       5. protected override void OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs e)
       6. {
       7. Stream playlistStream = Application.GetResourceStream(new Uri("Misc/Playlist.xml", UriKind.Relative)).Stream;
       9. System.Xml.Serialization.XmlSerializer serializer = new System.Xml.Serialization.XmlSerializer(typeof(Playlist));
       10. ActivePlaylist = (Playlist)serializer.Deserialize(playlistStream);
       12. using ( IsolatedStorageFile isoStorage = IsolatedStorageFile.GetUserStoreForApplication() )
       13. {
       14. using ( IsolatedStorageFileStream file = isoStorage.OpenFile("playlist.xml", FileMode.OpenOrCreate) )
       15. {
       16. var writer = new StreamWriter(file);
       18. serializer.Serialize(writer, ActivePlaylist);
       19. }
       20. }
       22. base.OnNavigatedTo(e);
       23. }
       24. ...
       25. }
   12. Trong constructor của trang này, thêm vào đoạn code như bên dưới để đăng kí cho sự kiện **PlayStateChanged.**
       1. C#
       2. public partial class MainPage : PhoneApplicationPage
       3. {
       4. ...
       5. public MainPage()
       6. {
       7. InitializeComponent();
       9. BackgroundAudioPlayer.Instance.PlayStateChanged += new EventHandler(Instance\_PlayStateChanged);
       10. }
       11. ...
       12. }
   13. Thêm hàm điều khiển sự kiện đã được đăng kí trong bước vừa rồi cùng với một số phương thức bổ sung cho hàm này. Các phương thức này có tác dụng cập nhật số track của track đang được phát bất kể trình phát âm thanh nền, cái mà chúng ta dùng để phát lại track, thay đổi trang thái của track đó.
       1. C#
       2. public partial class MainPage : PhoneApplicationPage
       3. {
       4. ...
       5. private void Instance\_PlayStateChanged(object sender, EventArgs e)
       6. {
       7. UpdateSelection();
       8. }
       9. private void UpdateSelection()
       10. {
       11. int activeTrackNumber = GetActiveTrackIndex();
       13. if ( activeTrackNumber != -1 )
       14. {
       15. lstTracks.SelectedIndex = activeTrackNumber;
       16. }
       17. }
       19. private int GetActiveTrackIndex()
       20. {
       21. int track = -1;
       22. if ( null != BackgroundAudioPlayer.Instance.Track )
       23. {
       24. track = int.Parse(BackgroundAudioPlayer.Instance.Track.Tag);
       25. }
       27. return track;
       28. }
       29. ...
       30. }
   14. Trình phát âm thanh nền sẽ cho phép người dùng thay đổi track đang được chơi trong khi ứng dụng không hoạt động. Để điều khiển chức năng này, hãy cập nhật phương thức **lstTracks\_Loaded** bằng cách chèn đoạn code sau vào phương thức này :
       1. C#
       2. public partial class MainPage : PhoneApplicationPage
       3. {
       4. ...
       5. private void lstTracks\_Loaded(object sender, RoutedEventArgs e)
       6. {
       7. UpdateSelection();
       8. }
       9. ...
       10. }
   15. Biên dịch và chạy ứng dụng, bạn sẽ được như sau :
       * 1. 
         2. Hình 2
         3. Playlist
   16. Stop debug và trở lại phần code.
   17. Kết thúc task 1.

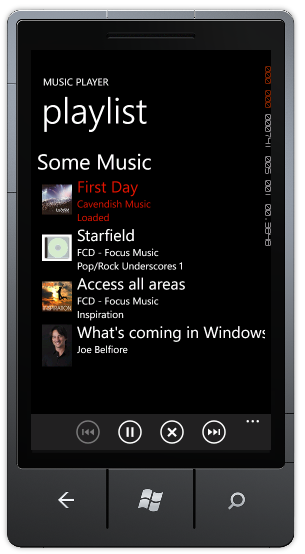
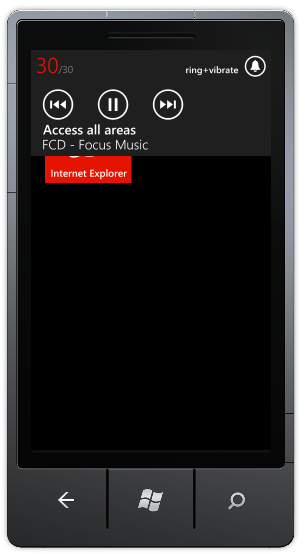
## Task 2 – Thêm các nút điều khiển

* 1. Task này sẽ hướng dẫn cách thêm các nút **ApplicationBar** dùng để điều khiển trình phát nhạc. Chúng ta sẽ thêm các nút sau : play, stop, previous and next.
  2. Chức năng của các nút này sẽ sử dụng các service được trình bày bởi lớp **BackgroundAudioPlayer.**
  3. Mở file **MainPage.xaml** nếu file này chưa được mở.
  4. Ngay sau khi kết thúc grid có tên **LayoutRoot**, ta thêm vào đoạn code sau :
     1. XAML
     2. <phone:PhoneApplicationPage.ApplicationBar>
     3. <shell:ApplicationBar IsVisible="True" IsMenuEnabled="False">
     4. <shell:ApplicationBarIconButton IconUri="/Images/prev.png" Text="prev" Click="appbar\_prev"/>
     5. <shell:ApplicationBarIconButton IconUri="/Images/play.png" Text="play" Click="appbar\_playpause"/>
     6. <shell:ApplicationBarIconButton IconUri="/Images/stop.png" Text="stop" Click="appbar\_stop"/>
     7. <shell:ApplicationBarIconButton IconUri="/Images/next.png" Text="next" Click="appbar\_next"/>
     8. </shell:ApplicationBar>
     9. </phone:PhoneApplicationPage.ApplicationBar>
  5. Mở fie **MainPage.xaml.cs**  và thêm vào các để xử lý sự kiện cho các nút vừa mới được thêm vào ở bước trên.
     1. C#
     2. public partial class MainPage : PhoneApplicationPage
     3. {
     4. ...
     5. #region ApplicationBar Buttons Events
     6. private void appbar\_prev(object sender, EventArgs e)
     7. {
     8. BackgroundAudioPlayer.Instance.SkipPrevious();
     9. }
     11. private void appbar\_playpause(object sender, EventArgs e)
     12. {
     13. if ( BackgroundAudioPlayer.Instance.PlayerState == PlayState.Playing )
     14. BackgroundAudioPlayer.Instance.Pause();
     15. else
     16. BackgroundAudioPlayer.Instance.Play();
     17. }
     19. private void appbar\_stop(object sender, EventArgs e)
     20. {
     21. BackgroundAudioPlayer.Instance.Stop();
     22. }
     24. private void appbar\_next(object sender, EventArgs e)
     25. {
     26. BackgroundAudioPlayer.Instance.SkipNext();
     27. }
     28. #endregion
     29. ...
     30. }
  6. Chúng ta chỉ mới cập nhật track được chọn ở task trước, bây giờ cần phải cập nhật thêm các nút AppBar tùy theo trạng thái của trình phát nhạc. Để làm điều này, trong phương thức **Instance\_PlayStateChanged**  ta gọi thêm phương thức **UpdateAppBarStatus.**
     1. C#
     2. public partial class MainPage : PhoneApplicationPage
     3. {
     4. ...
     5. private void Instance\_PlayStateChanged(object sender, EventArgs e)
     6. {
     7. UpdateAppBarStatus();
     8. UpdateSelection();
     9. }
     10. ...
     11. }
  7. Đây là phần nội dung của phương thức **UpdateAppBarStatus:**
     1. C#
     2. public partial class MainPage : PhoneApplicationPage
     3. {
     4. ...
     5. private void UpdateAppBarStatus()
     6. {
     7. switch ( BackgroundAudioPlayer.Instance.PlayerState )
     8. {
     9. case PlayState.Playing:
     10. //Prev Button
     11. if ( GetActiveTrackIndex() > 0 )
     12. ( ApplicationBar.Buttons[0] as ApplicationBarIconButton ).IsEnabled = true;
     13. else
     14. ( ApplicationBar.Buttons[0] as ApplicationBarIconButton ).IsEnabled = false;
     16. //Play/Pause Button
     17. ( ApplicationBar.Buttons[1] as ApplicationBarIconButton ).IsEnabled = true;
     18. ( ApplicationBar.Buttons[1] as ApplicationBarIconButton ).Text = "pause";
     19. ( ApplicationBar.Buttons[1] as ApplicationBarIconButton ).IconUri = new Uri("/Images/pause.png", UriKind.Relative);
     21. //Stop Button
     22. ( ApplicationBar.Buttons[2] as ApplicationBarIconButton ).IsEnabled = true;
     24. //Next button
     25. if ( GetActiveTrackIndex() < ActivePlaylist.Tracks.Count - 1 )
     26. ( ApplicationBar.Buttons[3] as ApplicationBarIconButton ).IsEnabled = true;
     27. else
     28. ( ApplicationBar.Buttons[3] as ApplicationBarIconButton ).IsEnabled = false;
     29. break;
     31. case PlayState.Paused:
     32. //Prev Button
     33. if ( GetActiveTrackIndex() > 0 )
     34. ( ApplicationBar.Buttons[0] as ApplicationBarIconButton ).IsEnabled = true;
     35. else
     36. ( ApplicationBar.Buttons[0] as ApplicationBarIconButton ).IsEnabled = false;
     38. //Play/Pause Button
     39. ( ApplicationBar.Buttons[1] as ApplicationBarIconButton ).IsEnabled = true;
     40. ( ApplicationBar.Buttons[1] as ApplicationBarIconButton ).Text = "play";
     41. ( ApplicationBar.Buttons[1] as ApplicationBarIconButton ).IconUri = new Uri("/Images/play.png", UriKind.Relative);
     43. //Stop Button
     44. ( ApplicationBar.Buttons[2] as ApplicationBarIconButton ).IsEnabled = true;
     46. //Next button
     47. if ( GetActiveTrackIndex() < ActivePlaylist.Tracks.Count - 1 )
     48. ( ApplicationBar.Buttons[3] as ApplicationBarIconButton ).IsEnabled = true;
     49. else
     50. ( ApplicationBar.Buttons[3] as ApplicationBarIconButton ).IsEnabled = false;
     51. break;
     53. case PlayState.Stopped:
     54. //Prev Button
     55. if ( GetActiveTrackIndex() > 0 )
     56. ( ApplicationBar.Buttons[0] as ApplicationBarIconButton ).IsEnabled = true;
     57. else
     58. ( ApplicationBar.Buttons[0] as ApplicationBarIconButton ).IsEnabled = false;
     60. //Play/Pause Button
     61. ( ApplicationBar.Buttons[1] as ApplicationBarIconButton ).IsEnabled = true;
     62. ( ApplicationBar.Buttons[1] as ApplicationBarIconButton ).Text = "play";
     63. ( ApplicationBar.Buttons[1] as ApplicationBarIconButton ).IconUri = new Uri("/Images/play.png", UriKind.Relative);
     65. //Stop Button
     66. ( ApplicationBar.Buttons[2] as ApplicationBarIconButton ).IsEnabled = false;
     68. //Next button
     69. if ( GetActiveTrackIndex() < ActivePlaylist.Tracks.Count - 1 )
     70. ( ApplicationBar.Buttons[3] as ApplicationBarIconButton ).IsEnabled = true;
     71. else
     72. ( ApplicationBar.Buttons[3] as ApplicationBarIconButton ).IsEnabled = false;
     73. break;
     75. case PlayState.Unknown:
     76. //Prev Button
     77. ( ApplicationBar.Buttons[0] as ApplicationBarIconButton ).IsEnabled = false;
     79. //Play/Pause Button
     80. ( ApplicationBar.Buttons[1] as ApplicationBarIconButton ).IsEnabled = true;
     81. ( ApplicationBar.Buttons[1] as ApplicationBarIconButton ).Text = "play";
     82. ( ApplicationBar.Buttons[1] as ApplicationBarIconButton ).IconUri = new Uri("/Images/play.png", UriKind.Relative);
     84. //Stop Button
     85. ( ApplicationBar.Buttons[2] as ApplicationBarIconButton ).IsEnabled = false;
     87. //Next button
     88. if ( GetActiveTrackIndex() < ActivePlaylist.Tracks.Count - 1 )
     89. ( ApplicationBar.Buttons[3] as ApplicationBarIconButton ).IsEnabled = true;
     90. else
     91. ( ApplicationBar.Buttons[3] as ApplicationBarIconButton ).IsEnabled = false;
     92. break;
     94. default:
     95. break;
     96. }
     97. }
     98. ...
     99. }
  8. Biên dịch và chạy ứng dụng, ta sẽ được như hình bên dưới :
     + 1. 
       2. Hình 3
       3. Playlist và các nút điều khiển.
  9. Stop debug và trở lại phần code.

## Task 3 – Copy các file nhạc và file ảnh vào Isolated Storage.

* 1. Để có thể phát được các track mà chúng ta có trong soluton, cần phải copy những file này vào **Isolated Storage.**
  2. Mở file có tên **App.xaml.cs**  và thêm vào các using sau :
     1. C#
     2. using System.IO.IsolatedStorage;
     3. using System.Windows.Resources;
  3. Trong nội dung contructors của lớp App, thêm vào dòng code sau :
     1. C#
     2. public App()
     3. {
     4. ...
     5. CopyToIsolatedStorage();
     6. }
  4. Nội dung của phương thức **CopyToIsolatedStorage** :
     1. C#
     2. private void CopyToIsolatedStorage()
     3. {
     4. using ( IsolatedStorageFile storage = IsolatedStorageFile.GetUserStoreForApplication() )
     5. {
     6. string[] files = new string[] { "file1.mp3", "file2.mp3", "file3.mp3" };
     8. foreach ( var \_fileName in files )
     9. {
     10. if ( !storage.FileExists(\_fileName) )
     11. {
     12. string \_filePath = "Music/" + \_fileName;
     13. StreamResourceInfo resource = Application.GetResourceStream(new Uri(\_filePath, UriKind.Relative));
     15. using ( IsolatedStorageFileStream file = storage.CreateFile(\_fileName) )
     16. {
     17. int chunkSize = 4096;
     18. byte[] bytes = new byte[chunkSize];
     19. int byteCount;
     21. while ( ( byteCount = resource.Stream.Read(bytes, 0, chunkSize) ) > 0 )
     22. {
     23. file.Write(bytes, 0, byteCount);
     24. }
     25. }
     26. }
     27. }
     29. files = new string[] { "Image1.jpg", "Image3.jpg", "no-art.jpg" };
     31. foreach ( var \_fileName in files )
     32. {
     33. string \_destFilePath = "Shared/Media/" + \_fileName;
     34. ;
     35. if ( !storage.FileExists(\_destFilePath) )
     36. {
     37. string \_filePath = "Images/" + \_fileName;
     38. StreamResourceInfo resource = Application.GetResourceStream(new Uri(\_filePath, UriKind.Relative));
     40. using ( IsolatedStorageFileStream file = storage.CreateFile(\_destFilePath) )
     41. {
     42. int chunkSize = 4096;
     43. byte[] bytes = new byte[chunkSize];
     44. int byteCount;
     46. while ( ( byteCount = resource.Stream.Read(bytes, 0, chunkSize) ) > 0 )
     47. {
     48. file.Write(bytes, 0, byteCount);
     49. }
     50. }
     51. }
     52. }
     53. }
     54. }
  5. Kết thúc task 3.

## Task 4 – Thêm và cài đặt Audio Playback Agent.

* 1. Việc cuối cùng chúng ta cần làm là cài đặt một “Audio Playback Agent”. Đối tượng này được tạo bởi hệ điều hành để điều khiển các action được yêu cầu bởi người dùng. **AudioPlayerAgent**  chạy ngầm và gọi vào một thể hiện của **BackgroundAudioPlayer**, sau đó gọi vào Zune Media Queue để phát âm thanh.
  2. Thêm mới một project vào solution sử dụng **Windows Phone Audio Playback Agent** template nằm trong catelory và đặt tên là **AudioPlaybackAgent.**
  3. Trong project này, chúng ra sẽ cần sử dụng một model được định nghĩa bên trong project **Models**  là XML serialization. Thêm references vào cả **Models** và **System.Xml.Serialization.**
  4. Mở file **AudioPlayer.cs** và xóa hết nội dung của lớp này.
  5. Thêm các namespace sau :
     1. C#
     2. using System.IO.IsolatedStorage;
     3. using System.Xml.Serialization;
     4. using System.IO;
     5. using Models;
  6. Thêm các biến static sau vào lớp **AudioPlayer.** Chúng sẽ được sử dụng để lưu playlist hiện tại và track đang được phát.
     1. C#
     2. static int currentTrackNumber = 0;
     3. static Playlist playlist;
  7. Thêm contructor sau, nó sẽ truy cập vào file XML playlist mà ta cần để đưa vào bộ nhớ isolated storage của ứng dụng.
     1. C#
     2. public AudioPlayer()
     3. : base()
     4. {
     5. //Load from IsoStore & deserialize
     6. using ( IsolatedStorageFile isoStorage = IsolatedStorageFile.GetUserStoreForApplication() )
     7. {
     8. using ( IsolatedStorageFileStream file = isoStorage.OpenFile("playlist.xml", FileMode.Open) )
     9. {
     10. XmlSerializer serializer = new XmlSerializer(typeof(Playlist));
     11. var reader = new StreamReader(file);
     12. playlist = (Playlist)serializer.Deserialize(reader);
     13. }
     14. }
     15. }
  8. Override phương thức **OnPlayStateChanged**, phương thức này sẽ xử lý sự thay đổi trạng thái của trình phát nhạc.
     1. C#
     2. protected override void OnPlayStateChanged(BackgroundAudioPlayer player, AudioTrack track, PlayState playState)
     3. {
     4. switch ( playState )
     5. {
     6. case PlayState.TrackEnded:
     7. PlayNext(player);
     8. break;
     9. case PlayState.TrackReady:
     10. player.Play();
     11. break;
     12. default:
     13. break;
     14. }
     15. NotifyComplete();
     16. }
     17. Bạn có thể thấy là một khi track được phát xong thì ta phải chuyển qua track kế tiếp ( sử dụng phương thức mà chúng tôi sẽ sớm trình bày) và một khi track nhạc đã sẵn sàng thì chúng ta chỉ đơn giản là sử dụng trình phát âm thanh nền để phát track đó. Phương thức **NotifyComplete**  được gọi để thông báo cho agent biết là sự thay đổi trạng thái đã được xử lý xong.
  9. Thêm các phương thức dùng để next và previous một track sau:
     1. C#
     2. private void PlayNext(BackgroundAudioPlayer player)
     3. {
     4. var songsCount = playlist.Tracks.Count;
     5. if ( ++currentTrackNumber >= songsCount )
     6. {
     7. currentTrackNumber = 0;
     8. }
     9. Play(player);
     10. }
     11. private void PlayPrev(BackgroundAudioPlayer player)
     12. {
     13. var songsCount = playlist.Tracks.Count;
     14. if ( --currentTrackNumber < 0 )
     15. {
     16. currentTrackNumber = songsCount - 1;
     17. }
     18. Play(player);
     19. }
  10. Override phương thức **OnUserAction**, phương thức này dùng để xử lý các sự kiện khác nhau do người dùng thực hiện khi khi tương tác với audio playback agent.
      1. C#
      2. protected override void OnUserAction(BackgroundAudioPlayer player, AudioTrack track, UserAction action, object param)
      3. {
      4. switch ( action )
      5. {
      6. case UserAction.FastForward:
      7. player.FastForward();
      8. break;
      9. case UserAction.Pause:
      10. player.Pause();
      11. break;
      12. case UserAction.Play:
      13. if ( player.PlayerState == PlayState.Paused )
      14. {
      15. player.Play();
      16. }
      17. else
      18. {
      19. Play(player);
      20. }
      21. break;
      22. case UserAction.Rewind:
      23. player.Rewind();
      24. break;
      25. case UserAction.Seek:
      26. player.Position = (TimeSpan)param;
      27. break;
      28. case UserAction.SkipNext:
      29. PlayNext(player);
      30. break;
      31. case UserAction.SkipPrevious:
      32. PlayPrev(player);
      33. break;
      34. case UserAction.Stop:
      35. player.Stop();
      36. break;
      37. default:
      38. break;
      39. }
      40. NotifyComplete();
      41. }
  11. Thêm phương thức **Play** **:**
      1. C#
      2. private void Play(BackgroundAudioPlayer player)
      3. {
      4. var currentTrack = playlist.Tracks[currentTrackNumber];
      5. Uri tileUri = ( currentTrack.Tile == null ? new Uri("Shared/Media/no-art.jpg", UriKind.Relative) :
      6. ( currentTrack.Tile.IsAbsoluteUri ? new Uri("Shared/Media/no-art.jpg", UriKind.Relative) :
      7. new Uri(currentTrack.TileString.Replace("/Images", "Shared/Media"), UriKind.Relative) ) );
      8. var audioTrack = new AudioTrack(currentTrack.Source,
      9. currentTrack.Title,
      10. currentTrack.Artist,
      11. currentTrack.Album,
      12. tileUri,
      13. currentTrackNumber.ToString(),
      14. EnabledPlayerControls.All);
      15. player.Track = audioTrack;
      16. }
  12. Override 2 phương thức **OnCancel** và **OnError** để xử lý những lỗi hoặc sự kiện hủy bỏ.
      1. C#
      2. protected override void OnCancel()
      3. {
      4. NotifyComplete();
      5. }
      6. protected override void OnError(BackgroundAudioPlayer player, AudioTrack track, Exception error, bool isFatal)
      7. {
      8. NotifyComplete();
      9. }
  13. Bước cuối cùng là ở project **MusicPlayer**, thêm liên kết đến project này ( tức là project **AudioPlayerAgent** ).
  14. Biên dịch ứng dụng và chắc chắn rằng bạn không có một lỗi nào.
  15. Chạy ứng dụng, lúc này trang chính của ứng dụng đã có đầy đủ chức năng.
  16. Và đây là lúc để phát nhạc. Click vào nút play () và chắc chắn là bạn đã mở âm thanh ( ấn F9 để tăng volume trong máy ảo).
      + 1. 
        2. Hình 4
        3. Ứng dụng đã có đầy đủ chức năng.
  17. Bây giờ bạn có thể thoát khỏi ứng dụng bằng nút Start Key () và nhận thấy rằng nhạc vẫn tiếp tục được phát.
  18. Nhấn F9 đê tăng volume sẽ làm hiện ra trạng thái của track hiện tại, và bạn có thể điều khiển phát phạt giống như đang ở trong giao diện chính của ứng dụng.
      1. 
      2. Hình 5
      3. Điều khiển trình phát nhạc khi ứng dụng đang chạy ngầm.
  19. Chuyển track hiện tại và chạy lại ứng dụng từ list các ứng dụng đã được cài đặt trong máy ảo.
  20. Ngừng debug.
  21. Đây là bước cuối cùng của task và cũng chính là bước kết thúc bài lab.

# Tổng kết

* 1. Bài lab này đã chỉ cho bạn cách sử dụng Windows® Phone Mango APIs để phát nhạc. bạn có thể nhận thấy rằng sử dụng API mới sẽ phát triển được một ứng dụng có thể phát nhạc ngay cả khi ứng dụng đó đang chạy ngầm.