[Listing 1]

public class QuotesDataContext : DataContext

{

private const string DBConnectionString = "Data Source=isostore:/Quotes.sdf";

public QuotesDataContext() : base(DBConnectionString) { }

public Table<Quote> Quotes;

}

[Listing 2]

[Table(Name="Quote")]

public class Quote : INotifyPropertyChanging

{

public event PropertyChangingEventHandler PropertyChanging;

string \_symbol;

[Column(IsPrimaryKey = true, IsDbGenerated=false)]

public string Symbol

{

get { return \_symbol; }

set {

if (\_symbol == value) return;

RaiseNotifyPropertyChanging("Symbol");

\_symbol = value;

}

}

string \_name;

[Column(Name = "Name")]

public string Name

{

get { return \_name; }

set

{

if (\_name == value) return;

RaiseNotifyPropertyChanging("Name");

\_name = value;

}

}

double? \_lastTradePriceOnly;

[Column(Name="LastTradePriceOnly")]

public double? LastTradePriceOnly

{

get { return \_lastTradePriceOnly; }

set

{

if (\_lastTradePriceOnly == value) return;

RaiseNotifyPropertyChanging("LastTradePriceOnly");

\_lastTradePriceOnly = value;

}

}

private void RaiseNotifyPropertyChanging(string propertyName)

{

if (PropertyChanging != null)

{

PropertyChanging(this, new PropertyChangingEventArgs(propertyName));

}

}

}

[Listing 3]

private void Application\_Launching(object sender, LaunchingEventArgs e)

{

using (var quotesDataContext = new QuotesDataContext())

{

if (!quotesDataContext.DatabaseExists())

{

quotesDataContext.CreateDatabase();

}

}

}

[Listing 4]

using (var quotesDataContext = new QuotesDataContext())

{

Quote quote = quotesDataContext.Quotes.SingleOrDefault(s => s.Symbol == "YHOO");

if (quote != null)

{

this.DataContext = quote;

}

}

[Listing 5]

using (var quotesDataContext = new QuotesDataContext())

{

quotesDataContext.Quotes.InsertOnSubmit(new Quote() { Symbol = "YHOO" });

quotesDataContext.SubmitChanges();

}

[Listing 6]

using (var quotesDataContext = new QuotesDataContext())

{

Quote quote = quotesDataContext.Quotes.SingleOrDefault(q => q.Symbol == "YHOO");

if (quote != null)

{

quote.Name = "Yahoo Inc";

quote.LastTradePriceOnly = 1.23;

}

quotesDataContext.SubmitChanges();

}

[Listing 7].

Quote quote = this.DataContext as Quote;

StandardTileData newTileData = new StandardTileData

{

Title = quote.Symbol,

BackContent = quote.Name.ToLower(),

BackTitle = quote.LastTradePriceOnly.ToString(new CultureInfo("fr-FR"))

};

if (quote.ChangeRealtime.HasValue && quote.ChangeRealtime.Value < 0)

{

newTileData.BackgroundImage = new Uri("/Assets/RedVariationIcon.png", UriKind.Relative);

newTileData.BackBackgroundImage = new Uri("/Assets/RedVariationIcon.png", UriKind.Relative);

}

else

{

newTileData.BackgroundImage = new Uri("/Assets/GreenVariationIcon.png", UriKind.Relative);

newTileData.BackBackgroundImage = new Uri("/Assets/GreenVariationIcon.png", UriKind.Relative);

}

ShellTile.Create(this.NavigationService.Source, newTileData);

[Listing 8]

ShellTile tileToUpdate = ShellTile.ActiveTiles.SingleOrDefault(t => t.NavigationUri.OriginalString == this.NavigationService.Source.OriginalString);

Quote quote = this.DataContext as Quote;

StandardTileData newTileData = new StandardTileData

{

Title = quote.Symbol,

BackContent = quote.Name.ToLower(),

BackTitle = quote.LastTradePriceOnly.ToString(new CultureInfo("fr-FR"))

};

if (quote.ChangeRealtime.HasValue && quote.ChangeRealtime.Value < 0)

{

newTileData.BackgroundImage = new Uri("/Assets/RedVariationIcon.png", UriKind.Relative);

newTileData.BackBackgroundImage = new Uri("/Assets/RedVariationIcon.png", UriKind.Relative);

}

else

{

newTileData.BackgroundImage = new Uri("/Assets/GreenVariationIcon.png", UriKind.Relative);

newTileData.BackBackgroundImage = new Uri("/Assets/GreenVariationIcon.png", UriKind.Relative);

}

tileToUpdate.Update(newTileData);

[Listing 9].

private void Application\_Launching(object sender, LaunchingEventArgs e)

{

[...]

PeriodicTask periodicTask = ScheduledActionService.Find("TileUpdateTask") as PeriodicTask;

if (periodicTask != null)

{

ScheduledActionService.Remove("TileUpdateTask");

}

try

{

periodicTask = new PeriodicTask("TileUpdateTask");

periodicTask.Description = "Tâche de mise à jour des Tiles";

ScheduledActionService.Add(periodicTask);

#if DEBUG

ScheduledActionService.LaunchForTest("TileUpdateTask", TimeSpan.FromSeconds(10));

#endif

}

catch (InvalidOperationException exception)

{

if (exception.Message.Contains("BNS Error: The action is disabled"))

{

MessageBox.Show("L'agent a été désactivé pour cette application.");

}

}

}

[Listing 10].

protected override void OnInvoke(ScheduledTask task)

{

try

{

if (task.Name == "TileUpdateTask")

{

// Code d'actualisation

[...]

}

}

catch (Exception ex)

{

// Gestion d'erreur

[...]

}

NotifyComplete();

}

[Listing 11].

<Tasks>

<DefaultTask Name="\_default" NavigationPage="MainPage.xaml" />

<ExtendedTask Name="BackgroundTask">

<BackgroundServiceAgent Specifier="ScheduledTaskAgent"

Name="Programmez.StockEchange.TaskAgent"

Source="Programmez.StockEchange.TaskAgent"

Type="Programmez.StockEchange.TaskAgent.ScheduledAgent" />

</ExtendedTask>

</Tasks>