Modul 6 : Pengunaan Library

In this tutorial we will use the following web service to convert currencies:
http://www.webservicex.net/CurrencyC...ToCurrency=EUR

There are several important aspects in this application.
This tutorial will only briefly touch each topic.

Files
You can add files to your project using the Files tab:

In our case we have two file. CountryCodes.txt is a text file containing the list of currencies. Each line contains exactly one value.
layout1.bal is the layout file created with the designer. Layout files are added automatically to the file manager.
Note that the layout file contains another two image files, the buttons arrows. These files are listed in the designer. If we remove layout1.bal they will be removed from the package as well.

The packaged files are also named assets. Locally they are stored under the Files sub folder.

This code reads the text file and stores the data in a list:
Code:

```
    If FirstTime Then
        countries = File.ReadList(File.DirAssets, "CountryCodes.txt")
```

File.ReadList is a convenient method that opens a file and adds all its lines to a List.
Files are always referenced by their folder and name.
The assets are referenced by the File.DirAssets value.
Android file system is case sensitive. Which means that image1.jpg is not the same as Image1.jpg (unlike Windows file system).

Structures

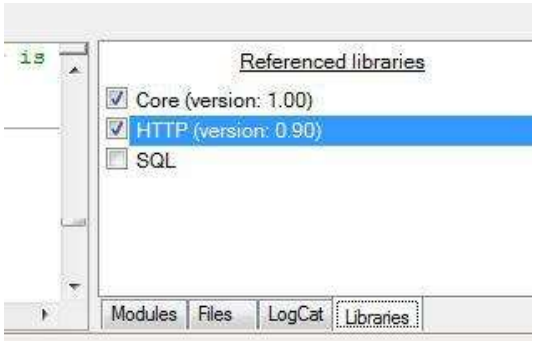
You can create new types or structures using the Type keyword. Later you can declare variables of these new types.
Types can hold any other objects, including other types and including themselves (and including arrays of all of these).
Structures will be covered more deeply in a different tutorial...
Structures are declared in one of the global subs.
Code:

```
Type MyTag (FromValue As EditText, ToValue As EditText, _
                  FromCurrency As Spinner, ToCurrency As Spinner)
    Dim CurrentTask As MyTag
```

This code declares a type that holds two EditTexts (textboxes) and two Spinners (Comboboxes).
We also declare a variable of that type named CurrentTask.
In the code you will see that we have another type named StateType which we use to store the current state.

All views have a property named Tag. You can set this property to any object you like. We are using it together with the Sender keyword to handle both buttons with the same sub.

Libraries



As you can see in the image, the Libraries tab page shows a list of available libraries. The checked libraries are referenced. Note that you cannot remove the reference to the core library.

Adding additional libraries

Libraries are made of a pair of files. The xml file that describes the library and the jar file which holds the compiled code.
Additional libraries and updates to official libraries are available here:
http://www.basic4ppc.com/forum/addit...icial-updates/
Note that only users who bought Basic4android can download these libraries.
To add a library to Basic4android all you need to do is to copy both files to a folder recognized by Basic4android.
By default this is the 'Libraries' folder that is usually located in: c:\Program Files\Anywhere Software\Basic4android\Libraries.
You can also configure an additional libraries folder by choosing Tools - Configure Paths. Note that the additional folder is scanned first for libraries. This means that you can update an existing library by putting the new files in the additional libraries folder (there is no need to delete the old files from the internal folder).

Http library

The Http library includes three objects.
HttpClient - This is the main object that executes and manages the requests and responses. The HttpClient can execute multiple requests concurrently.
It is very important to declare it as a Process global. The HttpClient handles requests that run in the background and it should not be tied to the activity life cycle.
Communication is done in two steps. First a connection is established by sending a HttpRequest object and then the response is read from the server.
The first step is always a non blocking action. It can take a long period till the connection

is established and you do not want to make your application be non-responsive at this time. Note that Android has a special "Application not responding" dialog which allows the user to force close the application.

The second step, which is the consumption of the response can be either blocking or nonblocking. If you download a file for example you should probably choose the nonblocking option.

This code creates and sends the GET request.
Code:

```
Dim request As HttpRequest
    request.InitializeGet(URL & fromCountry & "&ToCurrency="
& toCountry)
    request.Timeout = 10000 'set timeout to 10 seconds
    If HttpClient1.Execute(request, 1) = False Then Return 'Will
be false if their is already a running task (with the same id).
```

We are setting the timeout to 10 seconds which is quite short. The default is 30 seconds. The target web service is pretty unstable, which makes things more interesting. I prefer it to fail fast in our case.

HttpClient.Execute method receives two parameters. The first is the request object and the second is the Task ID. This integer will be passed back in the ResponseSuccess or ResponseError events.
It allows you to differentiate between different tasks that may be running in the background.
HttpClient.Execute will return false if their is already a running task with the same ID. This helps you prevent unnecessary multiple requests.
You can also check the status of running tasks with the IsBackgroundTaskRunning keyword.

Once the response is ready, ResponseSuccess or ResponseError will be raised. If things went well, we can now read the response, find the rate and display it. Otherwise we display a "toast" message with the error message.

As I wrote above, this specific web service seems to be unstable so your experience may vary.

State

As discussed in the life cycle tutorial we are required to save and restore the state of the application. In our case the state is made of the values in the text boxes and the current selected currencies.

The following type and variable are declared in Sub Process_Globals:
Code:

```
    Type StateType (TextUp As String, TextDown As String,
        _ IndexUp As Int, IndexDown As Int)
    Dim State As StateType 'This must be a process variable as
it stores the state
                           'and should not be released when the
activity is destroyed.
```

On the first run we set its values with the default values we want:
Code:

```
Sub ResetState
    'set the starting state
    State.TextUp = 1
    State.TextDown = ""
    State.IndexUp = 0 'USD
    State.IndexDown = 43 'Euro
End Sub
```

Later we save and read it as needed:
Code:

```
Sub Activity_Resume
    txtUp.Text = State.TextUp
    txtDown.Text = State.TextDown
    spnrUp.SelectedIndex = State.IndexUp
    spnrDown.SelectedIndex = State.IndexDown
End Sub

Sub Activity_Pause (UserClosed As Boolean)
    If UserClosed Then
        ResetState 'reset the state to the initial settings.
    Else
        State.TextUp = txtUp.Text
        State.TextDown = txtDown.Text
        State.IndexUp = spnrUp.SelectedIndex
        state.IndexDown = spnrDown.SelectedIndex
    End If
End Sub
```

In Activity_Resume we read the values and set the required views. Note that Activity_Resume is called right after Activity_Create. So it will also be called on the first time we run the application.

In Activity_Pause we save the value in the state object (which is a process variable). Note that if the user pressed on the back key (which means that he wants to close our application) we return the state to the initial state. Therefore the user will see a "clean new" application the next time he will run our application.

It is worth paying attention to this line:
Code:

```
CurrentTask.FromCurrency.SelectedItem.SubString2(0, 3)
```

CurrentTask is of type MyTag.
It has a field named FromCurrency which is of type Spinner.
Spinner has a property named SelectedItem which returns a String.
String has a method named Substring2.

Also note that this code is valid:
"abcd".Substring(2)
The complete code (file is also attached):
Code:

```
'Activity module

Sub Process_Globals
    Dim countries As List
    Dim URL As String
    URL =
"http://www.webservicex.net/CurrencyConvertor.asmx/ConversionRate?FromC
urrency="
    Dim HttpClient1 As HttpClient
    Type StateType (TextUp As String, TextDown As String,
         _ IndexUp As Int, IndexDown As Int)
    Dim State As StateType 'This must be a process variable as
it stores the state
                           'and should not be released when the
activity is destroyed.
End Sub

Sub Globals

    Dim txtUp, txtDown As EditText
    Dim spnrUp, spnrDown As Spinner
    Dim btnUp, btnDown As Button
    Type MyTag (FromValue As EditText, ToValue As EditText, _
                FromCurrency As Spinner, ToCurrency As Spinner)
    Dim CurrentTask As MyTag
End Sub
Sub ResetState
    'set the starting state
    State.TextUp = 1
    State.TextDown = ""
    State.IndexUp = 0 'USD
    State.IndexDown = 43 'Euro
End Sub

Sub Activity_Create(FirstTime As Boolean)

    If FirstTime Then
        Log("************************")
```

```
        'load the list of countries
        countries = File.ReadList(File.DirAssets, "CountryCodes.txt")
        'initialize the HttpClient object which is responsible for all
communication.
        HttpClient1.Initialize("HttpClient1")
        ResetState
    End If

    Activity.LoadLayout("layout1")

    spnrUp.AddAll(countries)
    spnrDown.AddAll(countries)

    Dim t1 As MyTag

    t1.FromValue = txtUp
    t1.ToValue = txtDown
    t1.FromCurrency = spnrUp
    t1.ToCurrency = spnrDown
    btnDown.Tag = t1

    Dim t2 As MyTag

    t2.FromValue = txtDown
    t2.ToValue = txtUp
    t2.FromCurrency = spnrDown
    t2.ToCurrency = spnrUp
    btnUp.Tag = t2
End Sub

Sub Activity_Resume

    txtUp.Text = State.TextUp
    txtDown.Text = State.TextDown
    spnrUp.SelectedIndex = State.IndexUp
    spnrDown.SelectedIndex = State.IndexDown
End Sub

Sub Activity_Pause (UserClosed As Boolean)

    If UserClosed Then
        ResetState 'reset the state to the initial settings.
    Else
        State.TextUp = txtUp.Text State.TextDown =
        txtDown.Text State.IndexUp =
        spnrUp.SelectedIndex state.IndexDown =
        spnrDown.SelectedIndex
    End If
End Sub
Sub btn_Click
    Dim btn As Button
    btn = Sender 'Fetch the actual button that raised this event.
    CurrentTask = btn.Tag 'Take the object from its Tag property.
    Dim fromCountry, toCountry As String
    fromCountry = CurrentTask.FromCurrency.SelectedItem.SubString2(0, 3)
'get the currency code
```

```
      toCountry = CurrentTask.ToCurrency.SelectedItem.SubString2(0, 3)

    Dim request As HttpRequest
    request.InitializeGet(URL & fromCountry & "&ToCurrency="
& toCountry)
    request.Timeout = 10000 'set timeout to 10 seconds
    If HttpClient1.Execute(request, 1) = False Then Return 'Will
be false if their is already a running task (with the same id).
    ProgressDialogShow("Calling server...")
End Sub
Sub HttpClient1_ResponseSuccess (Response As HttpResponse, TaskId
As Int)
    Log("ResponseSuccess")
    ProgressDialogHide
    Dim result As String
    result = Response.GetString("UTF8") 'Convert the response to
a string
    Log(result)
    Dim rate As Double
    'Parse the result
    i = result.IndexOf(".NET/")
    If i = -1 Then
        Msgbox("Invalid response.", "Error")
        Return
    End If
    i2 = result.IndexOf2("<", i + 1)
    rate = result.substring2(i + 7, i2)
    Log("Rate = " & rate)
    If IsNumber(CurrentTask.FromValue.Text) = False Then
        Msgbox("Please enter a valid number.", "Error")
        Return
    End If
    'Set the answer
    CurrentTask.ToValue.Text = Round2(CurrentTask.FromValue.Text *
rate, 2)
End Sub
Sub HttpClient1_ResponseError (Reason As String, StatusCode As
Int, TaskId As Int)
    Log(Reason)
    Log(StatusCode)
    ProgressDialogHide
    msg = "Error connecting to server."
    If reason <> Null Then msg = msg & CRLF &
    Reason ToastMessageShow (msg, True)
End Sub
```