

This article is archived because it is not considered relevant for third-party developers creating commercial solutions today. The article is believed to be still valid for the original topic scope.

Contents

- [1 Introduction](#)
- [2 XML Filter](#)
 - ◆ [2.1 Data Flow Diagram](#)
 - ◆ [2.2 Writing XML Filter](#)
- [3 Filter Explanation](#)
- [4 Character Encoding](#)
- [5 Code Snippet](#)
- [6 See Also](#)

Introduction

In [WidSets for Intermediate EP 1 : HTTP Request](#), I showed you how to download data from web server via http request. This page, I will show you how to work with data in XML file that has been downloaded via http.

XML Filter

XML is standard file format that is very popular for data exchange over the internet, for example, RSS feed. So, some develop may need to display data in these XML files on his/her widget. How to do that? ... You have 2 choices.

1. Download XML file via HTTP request and parse it yourself.
2. Download XML file via HTTP request and let XML Filter parse it for you.

Sure! I suggest you to select the second choice! Let?s see how it works.

Data Flow Diagram

In normal HTTP request, downloaded content will be sent to **onSuccess** callback function directly (in this example, it's **ok(...)** function).



But for HTTP request that cooperate with XML filter, downloaded content will be parsed by XML filter first and then the parsed data will be sent to **onSuccess** function afterward in the Value form.



Writing XML Filter

XML Filter is quite complicate. So let's learn by example. Here is the sample XML file.

data.xml

```

<?xml version="1.0" encoding="utf-8"?>
<userdata>
<name>Sittiphol Phanvilai</name>
<namethai>???????? ????</namethai>
<address>
<city>Bangkok</city>
<state>-</state>
<country>Thailand</country>
</address>
<pet type="Dog" name="Richie"/>
</userdata>
  
```

How to get data inside these tags? You have to modify **widget.xml** and add XML filter to parse this XML.

widget.xml

To use HTTP with XML filter, you have to **add http service with filter tag in widget.xml**.

```

<services>
  <service type="http" id="httpService">
    <filter id="digg"/>
  </service>
</services>
  
```

Next, write the XML filter in widget.xml with the name declared in service tag.

```

<filters>
  <filter id="digg">
    <list>
      <item name="name">
  
```

WidSets_for_Intermediate_EP_2 :_HTTP_with_XML_Filter

```
<xpath>/userdata/name/text ()</xpath>
</item>

<item name="namethai">
  <xpath>/userdata/namethai/text ()</xpath>
</item>

<item name="address">
  <choice>
    <xpath>/userdata/address/province/text ()</xpath>
    <xpath>/userdata/address/country/text ()</xpath>
    <xpath>/userdata/address/city/text ()</xpath>
  </choice>
</item>

<item name="pet">
  <strcat>
    <xpath>/userdata/pet/@name</xpath>
    <str> (</str>
    <xpath>/userdata/pet/@type</xpath>
    <str>)</str>
  </strcat>
</item>
</list>
</filter>
</filters>
```

httpxmlfilter.he

Write data fetching using HTTP request using **call** function.

```
void fetchxmldata()
{
  String URL = "http://www.nuuneoi.com/neoi/widsets/httpxmlfilter/data.xml";

  Prompt prompt = new Prompt(null, "Loading...", null, null);
  prompt.push();

  //widsets http service parameters
  Value arg = [
    "url" => URL
  ];
  call(null, "httpService", "get", arg, ok, nok);

  void ok(Object state, Value ret)
  {
    prompt.pop();
    prompt = null;
    flow.clear();
    foreach (Value item : ret) {
      String key = item[0];
      String value = item[1];
      {
        Label label = new Label(getStyle("labelKey"), key);
        label.setPreferredWidth(-100);
        label.setFlags(VISIBLE|LINEFEED);
        flow.add(label);
      }
      {
        Label label = new Label(getStyle("labelValue"), value);
        label.setPreferredWidth(-80);
      }
    }
  }
}
```

```

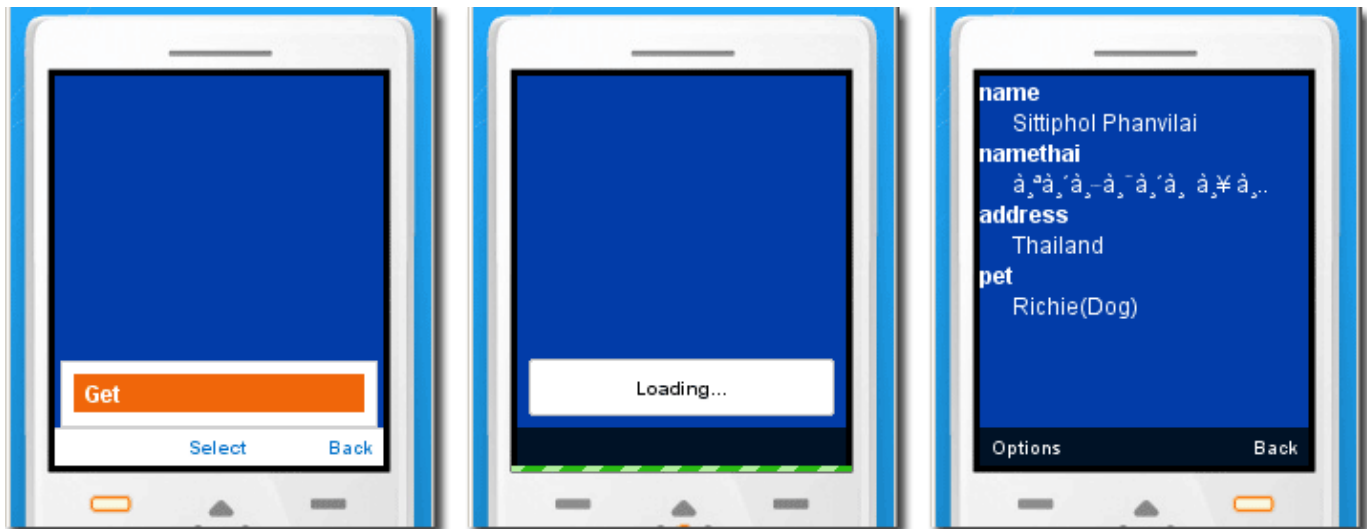
        label.setFlags(VISIBLE|LINEFEED);
        flow.add(label);
    }
}
setBubble(null, "Completed");
}

void nok(Object state, String error)
{
    prompt.pop();
    prompt = null;
    setBubble(null, "HTTP POST Failed: " + error);
}
}
}

```

When data get ready, function **ok** will be called with parsed data contain in **ret**. You can use **foreach** to get these value and use them.

Result



Filter Explanation

To get data inside tag, filter has to be written in this form.

```

<item name="KEY_NAME">
    <xpath>/path/to/data/text ()</xpath>
</item>

```

To get parameter value inside tag, filter has to be written in this form.

```

<item name="KEY_NAME">
    <xpath>/path/to/tag/@parametername</xpath>
</item>

```

For example:

```

<item name="name">

```

WidSets_for_Intermediate_EP_2 :_HTTP_with_XML_Filter

```
<xpath>/userdata/name/text () </xpath>
</item>
```

Here is data that has been sent to onSuccess function.

- key = **name**
- value = /userdata/name/text() which mean data in <userdata><name>...</name></userdata>

Filter can be written in many ways, <choice> tag can be applied to make more advance filter. <choice> will select **first data of XML nodes listed inside that is not null**.

```
<item name="address">
  <choice>
    <xpath>/userdata/address/province/text () </xpath>
    <xpath>/userdata/address/country/text () </xpath>
    <xpath>/userdata/address/city/text () </xpath>
  </choice>
</item>
```

- key = **address**
- value = first data of XML node listed below that is not null
 - ◆ <userdata><address><province>...</province></address></userdata>
 - ◆ <userdata><address><country>...</country></address></userdata>
 - ◆ <userdata><address><city>...</city></address></userdata>

You also can format string using <strcat> tag. All strings inside will be concatenated.

```
<item name="pet ">
  <strcat>
    <xpath>/userdata/pet/@name</xpath>
    <str>(</str>
    <xpath>/userdata/pet/@type</xpath>
    <str>)</str>
  </strcat>
</item>
```

- key = **pet**
- value will be formatted in **@name(@type)** .
 - ◆ /userdata/pet/@name means parameter value **name** in <userdata><pet ...> tag

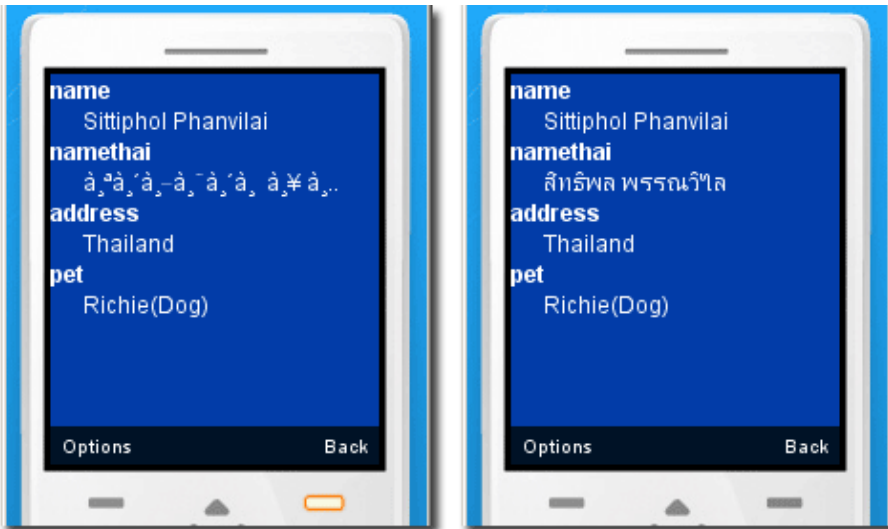
These are basic of XML Filter. If you need more advance XML filter, you can read at [Advanced filters](#).

Character Encoding

You may already notice that value in field **namethai** is not like one in XML. It happened because of character encoding problem. Data come in UTF-8 format but they are shown on screen using ASCII format. To solve this, you have to transcode data from ISO-8859-1 to UTF-8 using `decodeString` and `encodeString` method like this.

```
String key = decodeString(encodeString(item[0], "ISO-8859-1"), "UTF-8");
String value = decodeString(encodeString(item[1], "ISO-8859-1"), "UTF-8");
```

Result



Code Snippet

You can download source code for this tutorial from [File:WidSets HTTP with XML Filter Example.zip](#)

See Also

- [WidSets for Rookie EP 1 : First Step to WidSets SDK](#)
- [WidSets for Rookie EP 2 : First Compilation with WidSets SDK](#)
- [WidSets for Rookie EP 3 : Understand Hello World](#)
- [WidSets for Rookie EP 4 : Fasten WidSets Development](#)
- [WidSets for Rookie EP 5 : EditPlus Integration](#)
- [WidSets for Rookie EP 6 : Softkey Menu](#)
- [WidSets for Rookie EP 7 : Standard UI](#)
- [WidSets for Rookie EP 8 : Canvas](#)
- [WidSets for Rookie EP 9 : Timer](#)
- [WidSets for Rookie EP 10 : Key Handling](#)
- [WidSets for Intermediate EP 1 : HTTP Request](#)
- **[WidSets for Intermediate EP 2 : HTTP with XML Filter](#)**
- [WidSets for Advance EP 1 : Life Pictures Project](#)
- [WidSets SDK Tips : Emulator Language Changing](#)
- [WidSets SDK Tips : Emulator Skin Changing](#)
- [WidSets SDK Tips : Add Custom Emulator Skin](#)