Introduction to the settings.xml layout

sw6b, 20-04-2011 – covers settings.xml for the sw6.admin module

As covered in sw6.lib.1.0.documentation, to insert new settings into the administration module, a **`settings.xml`** file must be written and placed the **`/assets`** folder of the application. This document describes the legal elements and attributes of **`settings.xml`**. This document expects the developer to have basic knowledge of reading Document Type Definitions (DTD) for XML files – useful articles about this can be found at: W3C School (DTD) and Javacommerce (DTD elements).

Example

An example of how to structure the **`settings.xml`** file can be found here:
http://code.google.com/p/sw6android/source/browse/sw6.xmlvalidator/trunk/resources/settings.xml

Layout Documentation

Encoding

```
<?xml version = "1.0" encoding="UTF-8" ?>
```

According to this W3C article it is best practice to use the encoding attribute. Therefore: remember to save **`settings.xml`** in the same encoding as specified in the encoding attribute. We recommend UTF-8.

Doctype

```
<!DOCTYPE settings PUBLIC "-//SW6B//DTD settings.xml//sw6.xmlvalidator"
"http://sw6android.lcdev.dk/sw6b.xmlvalidator/settings.dtd">
```

The DTD is marked for public distribution. The URL indicates that we are using an external DTD located at some server.

Settings

```
Element  settings
DTD      <!ELEMENT settings (hidden?,visible?)>
```

The root element of the file.

Hidden

```
Element  hidden
DTD      <!ELEMENT hidden
         (boolean|double|integer|string|object|stdobject|enum)*>
```

An element that may contain a range of other elements as specified in the DTD. All elements specified inside this element will **not** be visible to the user on the phone or the PC interface.

Visible

```
Element  visible
DTD      <!ELEMENT visible
         (boolean|double|integer|string|object|stdobject|enum|menu)*>
```

An element that may contain a range of other elements as specified in the DTD. All elements specified inside this element **will be visible** to the user on the phone and the PC interface.

Menu

```
Element  menu
DTD      <!ELEMENT menu
         (boolean|double|integer|string|object|stdobject|enum|menu)+>
         <!ATTLIST menu title CDATA #REQUIRED>
         <!ATTLIST menu desc CDATA  #IMPLIED>
```

A menu must contain at least one element of the specified types. A menu may therefore contain submenus. Menus may be nested in as many levels as needed. A title must be provided for a menu. A description could be seen as a descriptive subtitle to the menu.

Enum and Elements

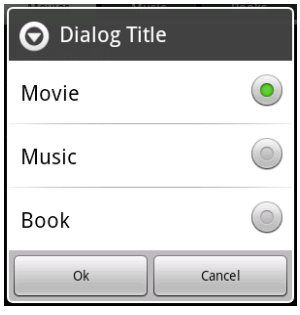
```
Element  enum
DTD      <!ELEMENT enum (element)+>
         <!ATTLIST enum varName CDATA  #REQUIRED>
         <!ATTLIST enum realName CDATA #IMPLIED>
         <!ATTLIST enum desc CDATA     #IMPLIED>
```

An enum must contain at least one element (pr. definition).

```
Element  element
DTD      <!ELEMENT element EMPTY>
         <!ATTLIST element realName CDATA #REQUIRED>
         <!ATTLIST element value    CDATA #REQUIRED>
```

An element represents a member of an enum.

Example of use



As Android support dialogs with select options, the enum element could be used to represent the data of such a dialog. See the picture.

To generate the menu shown, the enum must be placed inside a **<visible>** tag. The code to generate the menu would look like this:

```
<enum varName="entertainment" realName="Dialog Title">
          <element realName="Movie" value="6" />
          <element realName="Music" value="3" />
          <element realName="Book"  value="9" />
</enum>
```

As “Movie” is placed as the first element of the enum, this element will be selected as default.

Boolean

```
Element  boolean
DTD      <!ELEMENT boolean          (#PCDATA)>
         <!ATTLIST boolean varName  CDATA #REQUIRED>
         <!ATTLIST boolean realName CDATA #IMPLIED>
         <!ATTLIST boolean desc     CDATA #IMPLIED>
```

A setting representing a value of type **`boolean`**. A default value is optional. Valid default values are: **`true`** and **`false`**.

Double

```
Element  double
DTD      <!ELEMENT double           (#PCDATA)>
         <!ATTLIST double varName   CDATA #REQUIRED>
         <!ATTLIST double realName  CDATA #IMPLIED>
         <!ATTLIST double desc      CDATA #IMPLIED>
         <!ATTLIST double min       CDATA #IMPLIED>
         <!ATTLIST double max       CDATA #IMPLIED>
```

A setting representing a value of type **`double`**. A default value is optional. Minimum and maximum values may be defined as doubles.

Integer

```
Element  integer
DTD      <!ELEMENT integer          (#PCDATA)>
         <!ATTLIST integer varName  CDATA #REQUIRED>
         <!ATTLIST integer realName CDATA #IMPLIED>
         <!ATTLIST integer desc     CDATA #IMPLIED>
         <!ATTLIST integer min      CDATA #IMPLIED>
         <!ATTLIST integer max      CDATA #IMPLIED>
```

A setting representing a value of type **`integer`**. A default value is optional. Minimum and maximum values may be defined as integers.

String

```
Element  string
DTD      <!ELEMENT string           (#PCDATA)>
         <!ATTLIST string varName   CDATA #REQUIRED>
         <!ATTLIST string realName  CDATA #IMPLIED>
         <!ATTLIST string desc      CDATA #IMPLIED>
         <!ATTLIST string min       CDATA #IMPLIED>
         <!ATTLIST string max       CDATA #IMPLIED>
```

A setting representing a value of type **`String`**. A default value is optional. Minimum and maximum values may be defined for the length of the string.

Object

```
Element  object
DTD      <!ELEMENT object                EMPTY>
         <!ATTLIST object varName        CDATA #REQUIRED>
         <!ATTLIST object realName       CDATA #IMPLIED>
         <!ATTLIST object desc           CDATA #IMPLIED>
         <!ATTLIST object settingActivity CDATA #REQUIRED>
         <!ATTLIST object settingPc      CDATA #REQUIRED>
         <!ATTLIST object type           CDATA #REQUIRED>
```

Represents any object, e.g. an instance of **`java.util.ArrayList`** or **`sw6.somepkg.SomeNiceClass`**. The type is used to allow objects to be represented in the administration module. The objects must implement the **`Serializable`** interface. As described in sw6.lib.1.0.documentation, when an application is installed, **`settings.xml`** is parsed, and settings are inserted into the administration database. In this procedure, all objects will get **`null`** as their default reference. Therefore, we highly recommend all developers to **note this** and take care of it in their applications.

`settingActivity` defines the activity that is started when a parent wants to edit the settings for a particular object. The activity must be implemented in the developers application, and should take care of how the object should be edited. (More info: see: sw6.lib.1.0.documentation and Creating Activities for Custom Objects)

`settingPc` defines the class-path to a Java PC-GUI plug-in class that is loaded on the PC-interface when the user wants to edit the settings for this particular object. (More info: see: sw6.lib.1.0.documentation)

Please note that we at the moment haven't defined the precise implementation of **`settingPc`**. This means that one should just write **`null`** in this attribute.

Example of use

```
<object
          varName              = "calendar"
          realName             = "Child Calendar"
          desc                 = "Edit the child's calendar entries"
          settingActivity      = "sw6.app.calendar.EditCalendarActivity"
          settingPc            = "sw6.app.calendar.EditCalendar"
          type                 = "sw6.app.calendar.Calendar" />
```

This object could represent a calendar object. In the administration module, a list entry will show this setting - probably among other settings. The title of the setting will be the **`realName`**, and a small text shown below the title of this entry will contain the description defined in **`desc`** – if one is provided (remember, **`desc`** is optional). If a parent wants edit this setting on the phone, the **`settingActivity`** is started. On the PC, the Java plug-in provided by **`settingPc`** will be loaded and shown.

Stdobject

Element	stdobject		
DTD	<!ELEMENT stdobject		EMPTY>
	<!ATTLIST stdobject	varName	CDATA #REQUIRED>
	<!ATTLIST stdobject	realName	CDATA #IMPLIED>
	<!ATTLIST stdobject	desc	CDATA #IMPLIED>
	<!ATTLIST stdobject	type	CDATA #REQUIRED>

Represents one of the objects provided by **sw6.lib.types**, e.g. **sw6.lib.types.Interval**. These kinds of objects do not require the developer to develop an activity or a PC-GUI plug-in to enable editing of the object. For these kinds of objects, the **sw6.admin** module will provide GUI's to represent the object on the phone as well as on the PC. However, as with **objects**, the **stdobjects** are not initialized when an application is installed. This means that all **stdobjects** will get **null** as their default reference.

Example of use

```
<stdobject
            varName     = "appActionAvailability"
            realName    = "Availability Restriction"
            desc        = "Restrict the time interval in hours for when..."
            type        = "sw6.lib.types.Interval"/>
```

In this example a developer could use an instance of the **sw6.lib.types.Interval** class to represent the hours of the day for when some functionality should be available in his application.

Bug Report / Feature Request

Please report bugs and feature requests using: http://code.google.com/p/sw6android/issues/list