Settings Framework API

Note: :This API is not part of the public SDK. It can be found in the SDK API Plug-in.

Settings Framework is an application used for viewing dynamic plug-ins. This will load the plug-ins dynamically using ECom Architecture. It is designed to handle the minimum amount of plug-in related logic. All the plug-ins are implemented based on the ECom Plug-in Architecture.

Description

Creation of a General Setting Plug-in can be done directly by implementing **CGSPluginInterface**, or indirectly from the **CGSBaseView**. The Plug-in View class can be inherited from above classes based on the requirement. Create a container level class that implements the visualization of the plug-in view. Container level classes should be created when the view is activated

and destroyed when the view is deactivated.

CGSBaseView:

This class provides the common functionality in CAknView and CGSPluginInterface classes. One can inherit this class and implement the common functionality.

The responsibilities of the view level classes are:

- Implementing the view functionality defined by CGSBaseView:

```
TUid Id() const; // from CAknView
```

This returns the current view id.

- Creating and adding a container to the view stack when the view is activated.

```
void DoActivateL(
        const TVwsViewId& aPrevViewId,
        TUid /* aCustomMessageId */,
        const TDesC8& /* aCustomMessage */); // from CAknView, incase of multiple views
```

This function should call the NewContainerL() where a new instance of the container is created and add the container to the control stack.

- Removing the container from the view stack and destroying it when the view is deactivated.

```
void DoDeactivate(); // from CAknView, incase of multiple views
```

This function should remove container from the control stack and delete it.

```
void ConstructL();
void NewContainerL();
```

Create an instance of Container here.

```
void HandleListBoxSelectionL(); // from MEikListBoxObserver
```

- Implementing the plug-in functionality defined by CGSPluginInterface.

```
void GetCaptionL( TDes& aCaption ); //from CGSPluginInterface
```

This function is used in getting caption of this plugin.

CGSBaseContainer:

This class provides the common functionality for the CCoeControl classes. So a container level class can derive from this to implement the common functionality.

The container level class responsibilities are:

- Implementing the view functionality defined by CGSBaseContainer:

```
void ConstructL();
```

- Call this API which will initialize the ListBox:

```
BaseConstructL( const TRect& aRect, TInt aResTitleId, TInt aResLbxId );
```

This API internally calls ConstructListBoxL() which constructs the ListBox.

(C) Copyright Nokia 2013. All rights reserved.

- Call this API which will update the changes in the contents in the ListBox:

```
void UpdateListBoxL();
```

Example project

The following is the link to the Example Project:

File:SettingFrameWork.zip

Known issues

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.