Windows and Dialogs

A Servoy Smart or Web Client always comes with a main window and programmatically new windows can be opened or dialogs can be shown on top of existing windows to display Forms.

In This Chapter

- Definitions
- Smart Client modal dialog behavior difference in Java 5 & Java 6
- Web Client compatibility
  - Differences
  - Ironing out the differences
- Windowing API overview

Definitions

Window definition

> *Windows are independent of each other and are only tied to the Client. Closing the Client will close all windows. When a Smart or Web Client is launched it will always open with a window, the so-called main window. Closing the main window will close the Client and all other windows associated with the Client.*
>
> *In the Smart Client each window occupies an entry on the system bar. The individual windows can be minimized, maximized and closed independently and can be moved in front of one another.*
>
> *In the Web Client a window can be either a separate browser window or an additional tab in the existing browser window, depending the type of browser and/or the browsers configuration. The usefulness of opening new windows in browsers however is limited, due to browser restrictions on opening new windows/tabs programmatically from within a page. Almost any browser nowadays comes out of the box with a popup blocker installed, preventing pages from automatically opening new windows/tabs.*

Dialog definition

> *Dialogs are bound to a parent, being the main window, another window or dialog. A dialog is always on top of its parent and automatically close if their parent is closed. At the top of a hierarchy of dialogs, there is always one window, being either the main window or an additional window opened at a later stage.*
>
> *In the Smart Client a Dialog is similar to a Window, with the exceptions that it does not support maximize and minimize operations, on top of the general restrictions applied to dialogs (tied to a parent, always on top of their parent and automatically closed when the parent closes). While a dialog is always op top of it's parent, in the Smart Client the dialog window can be moved around independently and thus be located in such a position as to not overlap it's parent.*
>
> *In the Web Client a Dialog is displayed inline in the browser window/tab that displays the window in the dialogs parent hierarchy.*

Dialog modality

> *Dialogs can be either modal or non-modal (also referred to as modeless). A modal dialog blocks access to all other dialogs and the window in it's parent hierarchy, whereas non-modal dialogs do not block the access.*
>
> *In the Smart Client, if run under Java 5, the modal dialog will block all windows and dialogs, not only the ones in it's parent hierarchy. See Smart Client modal dialog behavior difference in Java 5 & Java 6 for more info.*
>
> *In the Smart Client the call that shows the modal dialog is blocking, meaning that it will not continue until the modal dialog is closed. Only when the dialog is closed will the rest of the code in the function that showed the modal dialog be executed.*
>
> *In the Web Client the call that shows the modal dialog in non-blocking, meaning it will return immediately and the rest of the code in the function will continue to execute.*

In addition to the ability to open new windows and show dialogs displaying Forms, Servoy also comes with a Dialogs plugin to show standard dialogs. This plugin only has limited support for usage in the Servoy Web Client.

Smart Client modal dialog behavior difference in Java 5 & Java 6

Due to limitations in Java 5 with regards to the modality API of windows and dialogs, a modal dialog in the Smart Client that runs under Java 5 will block ALL windows and dialogs, whereas when running under Java 6 the modal dialog will only block the dialogs and the window in its parent hierarchy.

Through the Smart Client settings on the Servoy Admin page, it's possible to restrict Smart Client to only run on Java 6 (or higher, although Java 6 has not been released yet). See Smart Client Settings for more information on specifying the min/max Java version for Smart Clients.

Web Client compatibility

Differences

As described in the introduction above, there are differences between the Smart Client and the Web Client when it comes to Windows and Dialogs:

1. While the Web Client supports opening new windows, but usefulness of the functionality is limited, as most likely they will be blocked by a popup blocker of the browser
2. Dialogs in the Web Client are displayed inline in the browser window/tab that displays the window at the top of the parent hierarchy of the dialog. Due to the fact that the dialog is shown inline, it cannot be moved in such a way that it doesn't overlap it's parent window.
3. The call to show a modal dialog is non-blocking in the Web Client and blocking in the Smart Client
4. The dialogs plugin only has partial support for the Web Client

The first two items in the above list are due to technical restrictions in the browser, where item 2 is a design decision based on the restriction mentioned under item 1.
The reasons behind item 3 & 4 are of technical nature and are likely to be resolved in a future version of Servoy. For the time being, the differences in items 3 & 4 can be ironed out.

Ironing out the differences

The solution to item 1 & 2 is to not use windows in combination with the Web Client: due to the fact that popups were abused on the web, every browser now comes with popup blockers that are enabled by default. Due to this, the defacto standard for web applications in general is a single window interface.

For item 3 there are 2 solutions possible:

- Make sure the call to show the modal dialog is the last thing done in the code.
  This is a straight forward solution, but can complicate the programming modal
- Use Continuations when running in the Web Client to simulate the blocking behavior seen in the Smart Client.
  Continuations are a advanced feature of the JavaScript engine utilized by Servoy to execute the JavaScript business logic. The Continuation functionality is only available when the business logic is executed on the Application Server, so for example in case of the Web Client. Continuations allow the pausing of code execution and resuming it at a later stage.

For item 4 the solution would be to use Forms diplayed as modal dialogs to mimic the functionality of the dialogs plugin, but this would be default have the same limitation mentioned under item 3. This could be countered using Continuations again.

The Continuations option is by far the most elegant solution and on ServoyForge there is a complete replacement for the dialogs plugin and the modal dialog API available in the form of a Servoy module that uses the Continuations API to provide the exact same behavior in the Smart and Web Client. To quote the project:

> *When using a dialog or a form in dialog being modal in web client then your method that opened this dialog doesn't wait until the (modal) dialog is closed like it would in Smart client.*
> *This module solves this issue (using JavaScript Continuations) and is in fact a full dialog plugin replacement.*
> *Apart from replacing the dialog plugin calls it provides a way to show modal form in dialogs that act the same way in web client.*
>
> *In short: when using this module your solution's dialogs and modal FID's are Web-client ready and work just the same in Smart-client.*

The Dialogs module on ServoyForge is a drop-in replacement for the dialogs plugin that ships with Servoy and for the modal dialog API of Servoy.

Windowing API overview

The API for windows and dialogs is located in several locations throughout the scripting API of Servoy:

- The JSWindow class: the main class for managing windows and dialogs displaying Forms in Servoy
- plugins.window: a plugin that provides access to the MenuBar, ToolBars and Status bar in the Smart Client and allows to create custom keyboard shortcuts in both the Smart and Web Client
- plugins.dialogs: a plugin that provides functions to show a number of simple dialogs. This plugin is only supported in the Smart Client
- The controller object of Forms:
  - controller.getFormContext()
  - controller.getWindow()
  - controller.show([windowName])
  - controller.showRecords(data, [windowname])
- The Application class:
  - application.closeAllWindows()
  - application.createWindow()
  - application.getScreenHeight()
  - application.getScreenWidth()
  - application.getWindow()
  - application.isFormIndialog()
  - application.showForm???
  - application.setToolbarVisible()