Appendix B - Usage of Methods Inherited from the Adapter Class

Appendix B - Usage of Methods Inherited from the Adapter Class

Inside the Application Designer management, adapters have to provide a defined interface to be managed correctly by the system. This interface is declared by `com.softwareag.cis.Server.IAdapter`. In order to have a high level of comfort during developing adapters, you should derive your adapter classes from the super-class `com.softwareag.cis.Server.Adapter`. This class already provides some useful methods.

This chapter covers the following topics:

- Access to Lookup Session Context
- Access to Application Designer Session Context
- Access to other Adapters
- Error Output
- Page Navigation
- Opening of Pop-up Dialogs
- Frame Communication
- Closing of a Page
- Multi Language Management

Access to Lookup Session Context

As you know, session management defines sessions (corresponding to one browser instance) and subsessions (corresponding to one process inside the Application Designer workplace). There is the possibility to bind and look for parameters on both levels:

- `Adapter.findSessionContext()` - returns the context which is on top of all subsessions. All adapters inside one session refer to the same session context.
- `Adapter.findSubSessionContext()` - returns the context which is held per subsession. Only adapters - belonging to the same subsession - share this context.

The result is a context supporting the interface `com.softwareag.cis.context.ILookupContext`. This interface provides two important methods:

```
public Object lookup(String s, boolean reactWithErrorIfNotExist);
public void bind(String s, Object o);
```

The session context is used, for example, to refer to the current user who is logged in, the chosen language, etc. The subsession context is used to share data inside a subsession.

Do not use the context as global variable buffers in a very intensive way. It will end up in programs relying on a lot of context information to be available - and sooner or later no one knows what has to be in the context when starting the program.

Via the methods

- `Adapter.findSessionId()`
- `Adapter.findSubsessionId()`

you can access the internally used representations of session ID and subsession ID.

Access to Application Designer Session Context

Application Designer uses its own lookup session management in order to store information of a session. You can access and manipulate this information by calling your adapter's method:

- `Adapter.findCISessionContext()` - returns a concrete session context object.

Inside the session context, the following parameters are kept:

- date format
- time format
- language
- style
- decimal separator
- and other information.

Have a look at the JavaDoc API documentation for more details.

Access to other Adapters

Access other adapters inside the same subsession by the methods:

- `Adapter.findAdapter(class)` - returns the adapter instance for a given class. Method `init()` is already called when passing back the instance - but only if the adapter was not used before.

Use this method before navigating between pages in order to prepare the adapter that will be used by the next page.

Error Output

You can display error messages inside the status bar (if it is defined in the page layout) by using the methods:

- `outputMessage(String, String (, String))`

First, pass a string for the type of message. This is needed to display a corresponding icon inside the status bar. There are constants defined inside the Adapter for specifying the type:

- `Adapter.MT_ERROR`
- `Adapter.MT_WARNING`
- `Adapter.MT_SUCCESS`

The second string is the message being shown.

The third string - which is optional - is the long text description of the message. It becomes visible by a dialog if the user clicks with the mouse on the message. If you do not specify a long description, the normal message is used.

Page Navigation

Navigate to a page by using the method:

- `switchToPage(String pageName)`

The "pageName" is the URL - either relative or absolute - of the next page.

Opening of Pop-up Dialogs

You can open a page inside a pop-up dialog by using the method:

- `openPopup(String pageName)`.

The "pageName" is the URL - either relative or absolute - of the page that is displayed inside the dialog.

You can specify pop-up parameters of the pop-up you open with `openPopup()` by using the methods:

- `setPopupTitle(String title)`
- `setPopupPageFeatures(String pageFeatures)`

Frame Communication

There are various methods to communicate to other frames:

- `openPageInTarget`

- `openCISPageInTarget`
- `invokeMethodInTarget`
- `refreshTarget`
- `sizeTarget`

Closing of a Page

The default method used for closing a page is `endProcess()`. It is provided by the `Adapter` class. The tasks performed by the `endProcess()` method are:

- The current subsession is closed and de-registered inside the session management.
- The current page is de-registered from the workplace management - if it was registered before.

Calling the `endProcess()` method ensures that all memory resources are released for the corresponding subsession.

The `endProcess()` method is called by clicking inside the page on the close icon at the top right corner of the page. You can also call it directly inside an adapter, e.g. if you want to close the subsession as reaction to the user's entered data.

Multi Language Management

You can access the multi language management using the methods:

- `replaceLiteral(String application, String textid)`
- `replaceLiteral(String application, String textid, String param1)`
- `replaceLiteral(String application, String textid, String param1, String param2)`
- `replaceLiteral(String application, String textid, String param1, String param2, Stirng param3)`

The application is the name for the abbreviation of a defined application area for which literals are defined. In the file-based multi language management, it represents the name of a CSV file that holds the text identified by a text ID.