Standard Pop-up Dialogs

There are standard pop-up dialogs available for general usage which you do not have to code yourself.

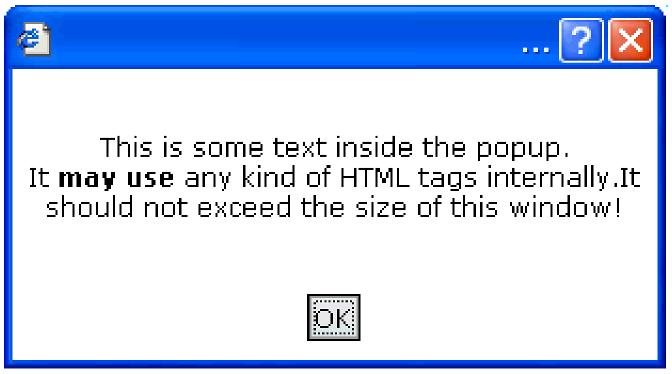
This chapter covers the following topics:

- OK Pop-up
- Yes/No Pop-up
- Log Pop-up
- Example: Asking Whether the User Really Wants to Quit

OK Pop-up

The OK pop-up is used for displaying a text with an **OK** button.

The following is an example of an OK pop-up:



This is some text inside the popup.
It **may use** any kind of HTML tags internally.It should not exceed the size of this window!

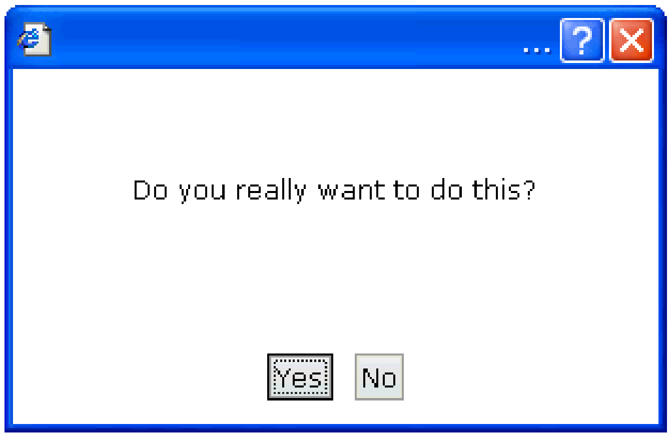
The code of the adapter is:

```
public void showOKPopup()
{
    PopupOKModel pok = (PopupOKModel)findAdapter(PopupOKModel.class);
    pok.init("This is some text inside the pop-up.<br>"+
             "It <b>may use</b> any kind of HTML tags internally." +
             "It should not exceed the size of this window!");
    this.openPopup("/HTMLBasedGUI/popupok.html");
}
```

Yes/No Pop-up

The Yes/No pop-up is used for asking the user a question. Depending on user’s decision, activities are started inside the adapter.

The following is an example of a Yes/No pop-up:



The code of the adapter is:

```
public class YESCommand implements com.softwareag.cis.server.util.ICommand
{
    public void execute()
    { outputMessage("S","Yes command was called"); }
}
public class NOCommand implements com.softwareag.cis.server.util.ICommand
{
    public void execute()
    { outputMessage("S","No command was called"); }
}
public void showYESNOPopup()
{
    PopupYesNoModel pyn = (PopupYesNoModel)findAdapter(PopupYesNoModel.class);
    pyn.init("Do you really want to do this?",
              new YESCommand(),
              new NOCommand());
    this.openPopup("/HTMLBasedGUI/popupyesno.html");
}
```

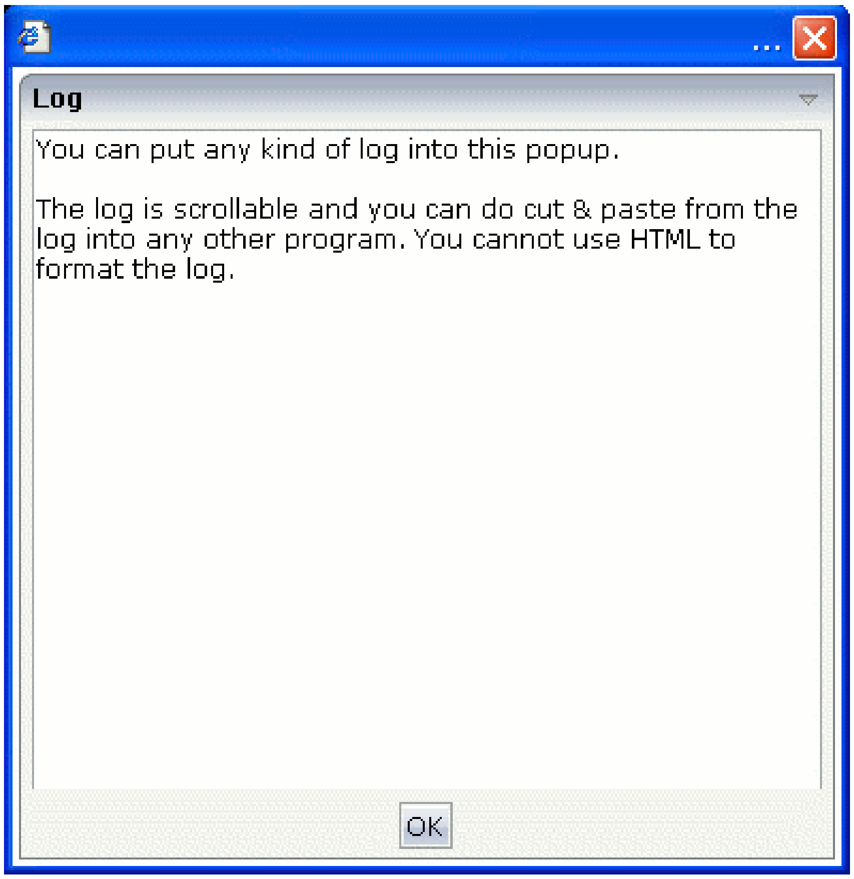
The pop-up dialog is initialised by passing the question and two "reaction objects" to it. One "reaction object" is called when choosing the **Yes** button, the other is called when choosing the **No** button.

The "reaction objects" have to implement the interface `com.softwareag.cis.server.util.Icommand` which just needs a simple `execute()` method. In our example, the "reaction objects" are implemented as inner classes of the adapter class.

Log Pop-up

The Log pop-up is used for displaying a log text.

The following is an example of a Log pop-up:



You can put any kind of log into this popup.

The log is scrollable and you can do cut & paste from the log into any other program. You cannot use HTML to format the log.

The code inside the adapter is:

```
public void showLOGPopup()
{
    PopupLogModel plm = (PopupLogModel)findAdapter(PopupLogModel.class);
    plm.init("You can put any kind of log into this pop-up.\n\n"+
             "The log is scrollable and you can do cut & paste from "+
             "the log into any other program. You cannot use HTML "+
             "to format the log.");
    this.openPopup("/HTMLBasedGUI/popuplog.html");
}
```

Example: Asking Whether the User Really Wants to Quit

This is a typical example: the user works on a page of your application for a while and then choose the close icon in the right top corner of the page. Check whether the user has changed something and ask using a pop-up dialog if the user really wants to close the page.

The following Java source shows an implementation in the adapter class:

```
/** */
public void endProcess()
{
    if (m_changed == true)
    {
        PopupYesNoModel pyn = (PopupYesNoModel)findAdapter(PopupYesNoModel.class);
        pyn.init("You modified some data. Do you really want to exit?",
                 new ICommand() { public void execute() { executeEndProcess(); }},
                 null);
        this.openPopup("/HTMLBasedGUI/popupyesno.html");
    }
    else
    {
        executeEndProcess();
    }
}

/** */
public void executeEndProcess()
{
    super.endProcess();
}
```

The `endProcess()` method is called by the closing function of the page. It is provided by the `Adapter` class from which the adapter is inherited. The `endProcess()` method does already everything which is required for removing the subsession.

Overwrite the `endProcess()` method and embed the code which opens a Yes/No pop-up to ask whether the user really wants to quit the application. The original closing function is shifted to the method `executeEndProcess()`. The Yes/No pop-up got for the "Yes" method an inner class pointing to the `executeEndProcess()` method. The "No" method is null and means that nothing should be done.