Building Own Workplaces as a Frameset Definition

Building Own Workplaces as a Frameset Definition

A set of functions is available which simplify the usage of Application Designer HTML pages inside a given HTML frameset definition. The functions are not only usable in the scope of workplace/portal management, but can also be used apart from this.

This chapter covers the following topics:

- Basics
- Defining the Frameset
- Simple Way of Opening Pages in Frames
- A More Complex Way of Opening Pages in Frames
- When to Use the Complex Way
- Opening Normal HTML Pages inside Frames
- Frame Communication
- Multiple Frame Operations
- When Building your Own Workplaces

Basics

The basic functions cover the following aspects:

- You can define an HTML page containing any kind of frameset you want. In this page, you design the frames, their sizes, their scroll behavior, their behavior when resizing the screen, etc. For each frame which which you want to interact, you define an identifier name.
- You open Application Designer pages inside the frames. There are two possibilities:
    1. Open these pages with a URL as described in the previous section.
    2. Open these pages with adapter methods (server-side processing).

This section will focus on the second possibility since the first is just a certain usage of what is described in the previous section. This offers you an explicit control about what happens inside the frames: e.g. a page within frame "A" should be replaced by another page. Before proceeding, the user should be asked whether to store unsaved data (or not).

It is possible to communicate with frames on the client side. This means, you can build up interaction (e.g. you want to update another frame’s content) without any flickering in the target frame.

Defining the Frameset

In the following screen, a page is shown which is divided into three frames:

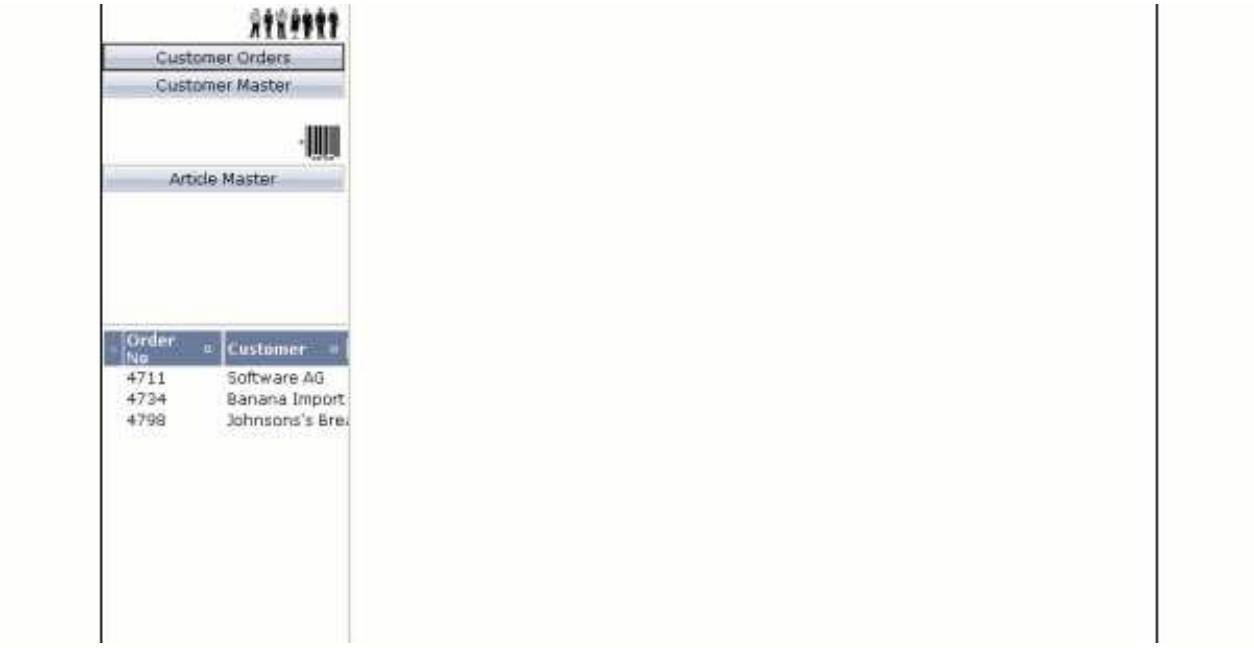
The corresponding frameset definition of the page is:

```
<html>

<head>
<title>New Page 2</title>
<meta name="GENERATOR" content="Microsoft FrontPage 4.0">
<meta name="ProgId" content="FrontPage.Editor.Document">
</head>

  <frameset cols="200,*">
    <frameset rows="*,*">
      <frame name="lefttop" src="/cis/servlet/StartCISPage?PAGEURL=/cisdemos/frameleft.html">
      <frame name="leftbottom" src="blank.html">
    </frameset>
    <frame name="right" src="blank.html">
  </frameset>

</html>
```

The frameset contains three frames with the IDs `lefttop`, `leftbottom` and `right`. The `lefttop` frame opens the Application Designer page */cisdemos/frameleft.html*. This page contains buttons for some functions and acts like a "menu page".

Simple Way of Opening Pages in Frames

When choosing the **Customer Orders** button, the corresponding Application Designer page is opened in the `leftbottom` frame:



The page shows a list of customer orders. It is a normal Application Designer page. How can it be opened by choosing the **Customer Orders** button?

The */cisdemos/frameleft.html* page (acting as a "menu page") is hooked on to a Java adapter class which looks as follows:

```
import com.softwareag.cis.server.Adapter;

// This class is a generated one.

public class FrameLeftAdapter
    extends Adapter
{
    /** */
    public void onArticleMaster()
    {
        // TODO Auto-generated method stub
    }

    /** */
    public void onCustomerMaster()
    {
        // TODO Auto-generated method stub
    }

    /** */
    public void onCustomerOrders()
    {
        this.openCISPageInTarget("OpenCustomerOrders.html", "leftbottom");
    }
}
```

By choosing the **Customer Orders** button, the method `onCustomerOrders` is called. This method performs a method `openCISPageInTarget` inherited from class `Adapter`. The first parameter of the method is the page that is to be opened; the second parameter defines the ID of the frame in which the page is to be opened.

The page *OpenCustomerOrders.html*, which is opened when choosing the **Customer Orders** button, is running inside the same subsession as the page from which it was called. If you need to access the page adapter before opening the page inside the "leftbottom" frame, use the `findAdapter` method inside your adapter.

A More Complex Way of Opening Pages in Frames

When selecting an order in the `leftbottom` area of the previous example, a customer order page is displayed in the right frame:

The data from the order you selected is transferred into the corresponding fields of the customer order page. Have a closer look at the details.

This is the source of the adapter for listing customer orders:

```
import java.util.Iterator;

import com.softwareag.cis.server.Adapter;
import com.softwareag.cis.server.IInteractionSessionMgr;
import com.softwareag.cis.server.InteractionSessionMgrFactory;
import com.softwareag.cis.server.util.SelectableLine;
import com.softwareag.cis.server.util.TEXTGRIDCollection;
import com.softwareag.cis.util.CDate;

public class OpenCustomerOrdersAdapter
    extends Adapter
{
    // ------------------------------------------------------------------------
    // inner classes
    // ------------------------------------------------------------------------

    public class Order
        extends SelectableLine
    {
        public Order(String number, String date, String customer)
        {
            m_customer = customer;
            m_date = new CDate(date);
```

```
        m_number = number;
    }

    // property >orders[*].customer<
    String m_customer;
    public String getCustomer() { return m_customer; }
    public void setCustomer(String value) { m_customer = value; }

    // property >orders[*].date<
    CDate m_date;
    public CDate getDate() { return m_date; }
    public void setDate(CDate value) { m_date = value; }

    // property >orders[*].number<
    String m_number;
    public String getNumber() { return m_number; }
    public void setNumber(String value) { m_number = value; }
}

// ---------------------------------------------------------------------
// property access
// ---------------------------------------------------------------------

// property >orders<
TEXTGRIDCollection m_orders = new TEXTGRIDCollection();
public TEXTGRIDCollection getOrders() { return m_orders; }

// ---------------------------------------------------------------------
// public adapter methods
// ---------------------------------------------------------------------

public void onOrderSelect()
{
    // find the selected item
    Order selectedOrder = null;
    Iterator iter = m_orders.iterator();
    while (iter.hasNext())
    {
        selectedOrder = (Order)iter.next();
        if (selectedOrder.getSelected() == true)
            break;
        else
            selectedOrder = null;
    }
    if (selectedOrder == null)
        return;
    // session management: "refresh" subsession
    String sessionId = this.m_interactionProcess.getSessionId();
    IInteractionSessionMgr iism = InteractionSessionMgrFactory.getInteractionSessionMgr();
    iism.removeSubsession(sessionId,"subsession_right");
    iism.createNewSubsession(sessionId,"subsession_right");
    // prefetch and manipulate adapter inside the refreshed subsession
    CustomerOrderDetailAdapter coda = (CustomerOrderDetailAdapter)iism.findAdapterInSubsession
        (sessionId, // sessionID
         "subsession_right", // subsessionId
         CustomerOrderDetailAdapter.class.getName(), // class
         "", // pageId, typically ""
         findPageApplication()); // application project
    coda.setNumber(selectedOrder.getNumber());
    coda.setName(selectedOrder.getCustomer());
    // navigate to page
    openCISPageInTarget("CustomerOrderDetail.html","subsession_right","right");
}

// ---------------------------------------------------------------------
// standard adapter methods
// ---------------------------------------------------------------------

// property >messageType< implemented in Adapter
// property >messageShortText< implemented in Adapter
```

```
    // property >messageLongText< implemented in Adapter

    /** initialisation - called when creating this instance*/
    public void init()
    {
        m_orders.add(new Order("4711","20020706","Software AG"));
        m_orders.add(new Order("4734","20020702","Banana Import Export Ltd."));
        m_orders.add(new Order("4798","20020604","Johnsons's Bread"));
    }
}
```

With method `onOrderSelect`, the selected line is determined first.

In the next steps, frame communication is prepared and finally done. The difference to the previous "simple" scenario is that the page which is opened runs in a different subsession inside the session management of Application Designer.

Remember that each browser instance internally requests one session, and that each session is divided into various subsessions. Adapters are running inside subsessions. The subsession is responsible for keeping and releasing resources. It corresponds to one interaction process which has a defined life cycle - e.g. the data input of a customer order. For more information, see the section *Session Management*. Each subsession has an identifier - in this example, the name of the subsession is `subsession_right`. You can also create a unique ID with the class `com.softwareag.cis.util.UniqueIdMgmt`.

Our example program first removes the subsession `subsession_right`. Everything which is currently managed inside the subsession will be released. Since there is no subsession when being called the first time, no error will occur.

After releasing this subsession, a new subsession is immediately created. With the interaction session manager, you can access a method which passes back an adapter instance inside a given subsession. Like the method `findAdapter` of class `Adapter`, this method returns an adapter object which is managed inside the same subsession in which the adapter is running. With the interaction session manager, you can also access adapters inside different subsessions.

The returned adapter instance gets the selected data. Finally, the frame communication takes place: pay attention that the ID of the subsession has to be passed inside the `openCISPageInTarget` method.

When to Use the Complex Way

The complex way should be your "standard thinking" in this scenario. When dealing with Application Designer pages inside different frames, you have to take care about how you manage your sessions at the server side.

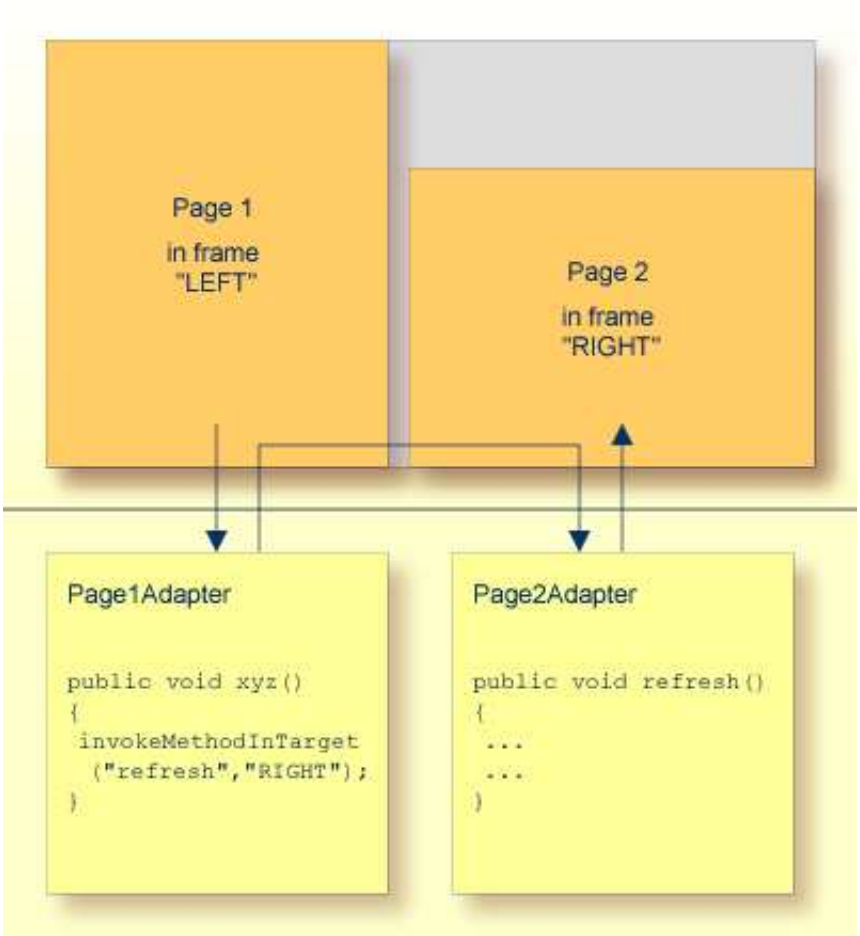
The content which runs inside the frames (e.g. the Customer Order screen) is not aware of these session management dependencies. But the designer of the workplace has to take care of the interaction possibilities inside the workplace.

Opening Normal HTML Pages inside Frames

In addition to the methods `openCISPageInTarget`, there is an equivalent method `openPageInTarget` which you inherit from the `Adapter` class. This page opens a normal HTML page inside one frame.

Frame Communication

When working with frames, it is possible to open a client-side communication channel between frames: by the client, you call an adapter method of an Application Designer page which is opened in a different frame. This is done by the method `invokeMethodInTarget(methodName,targetName)` which you inherit from the `com.softwareag.cis.server.Adapter` class.



This sounds strange at first: in the adapter processing of frame `LEFT`, you call an adapter method of frame `RIGHT` and the call is executed by the client - what is the reason for this? The advantage of calling the method by the client is that the call is initiated by the page which runs inside the target frame. This means that the target frame sends the request for method execution and updates its content as a reaction of the request's response. In other words: you can update pages in other frames without redrawing the page, i.e. without flickering.

This is a very powerful way of allowing communication between frames - but there are some restrictions which you have to keep in mind:

- The frame which initiates the interaction as well as the target frame must be in the same frameset page - in other words: they must share the same "document parent".
- The page shown in the target frame must be received by the same server and port as the page which initiates the communication. Otherwise, you will receive a JavaScript security exception - it is (by default) not allowed to establish a client communication between pages coming from different hosts.

The frame communication framework acts upon error situations in a quite tolerant way:

- If you invoke a method inside a frame target and the frame does not exist, nothing will be done on the client side.
- If you invoke a method inside a frame target and there is no valid Application Designer page inside the frame, nothing will be done.
- If you invoke a method inside a frame target and the corresponding "target adapter" does not provide the requested method, the content of the frame target’s page will be synchronized with its adapter. This is similar to defining a BUTTON control and specifying a method which does not exist inside the adapter.

Tip:
Only use frame communication if you want to update the content of another frame page. Do not use this method for "normal" interaction between adapters, without any changes on the page.

Multiple Frame Operations

You can call the methods `openCISPageInTarget`, `openPageInTarget` and `invokeMethodInTarget` multiple times inside one request, for example, if there are multiple frames you want to manipulate at the same time.

When Building your Own Workplaces

As you learned in this section, Application Designer provides powerful mechanisms to build flexible workplace/portal scenarios. Be aware that workplace management means more than bringing up some pages into different frames. Workplace management also means, for example, that you take care of opening and closing the session state (in Application Designer: subsession state) and that you have to provide global data (like the currently logged in user) shared by all adapters, etc.