Appendix A - Call Sequence for Adapter

Appendix A - Call Sequence for Adapter

This chapter describes how an incoming request by the browser client is processed inside an adapter. The request contains all the changes of properties that have been made at client side. The following topics are covered:

- Normal Call Sequence
- Call Sequence when a Subsession is Destroyed
- Call Sequence when a Session is Destroyed
- Error/ Runtime Exceptions
- Pay Attention when Overwriting

Normal Call Sequence

- **init()**
  This method is called only once - when creating the adapter inside a subsession. Before calling this method, Application Designer makes sure that the adapter instance is properly registered inside the Application Designer environment. Therefore - for example - you have access to the session management: use the `findSessionContext()` or `findSubSessionContext()` method in order to look for some values inside the `init()` method. It is not possible to use the `find...SessionContext()` methods inside the constructor of an adapter - since the session is not yet assigned to the adapter instance.

  When navigating between pages (using the `switchToPage()` or `openPopupPage()` method), the corresponding adapter objects are only created once. For example, if you navigate from page "A" to page "B" and back to page "A", the adapter of page "A" does not change. The `init()` method is only called once - at the time the adapter is instanciated.

- **activate(...)**
  This method is implemented by the `Adapter` class already. You only need to overwrite this method if you want to passivate the state between requests. In this case, you can activate this state inside your implemented method of your adapter class. If you use the adapter class to cooperate, for example, with EJB objects running inside the context of an EJB server, you should synchronize the state passivation with the EJB server's passivation.

- **reactOnDataTransferStart()**
  This method is called when the transfer of the changed properties starts. You can initialize some internal members at this time. If you overwrite this method, do not forget to include the method of the super-class (`Adapter.reactOnDataTransferStart()`) into your method implementation!

- **setXxx(), setYyy(), ...**
  Now, the set methods of the changed properties of the browser client are transferred. It is very important that your implemented set methods never cause an exception or an error.

- **reactOnDataTransferEnd()**
  This method is called after setting the changed properties. Use this method to perform operations you always want to execute when processing a request.

- **invoke()-Method**
  If the request has a method call inside, the method is invoked now.

- **processAsDefault()**
  If the request has no method call, this standard method is called.

- **reactOnDataCollectionStart()**
  This method is called when the transfer of adapter properties starts. Use this method, for example, for performance improvements during the following get methods, for example, by building temporary objects.

- **getXxx(), getYyy()**
  All get methods of the adapter - including array elements which may be passed back by - are called.

- **reactOnDataColletionEnd()**
  This method is called when data collection is finished. Temporary objects - which you may have created for performance reasons - can be released for garbage collection now.

- **passivate(...)**
  This method is the counterpart of the activate method.

Call Sequence when a Subsession is Destroyed

- **endProcess()**
  This method is called inside the adapter if the user decides to terminate the subsession. For example, in the Application Designer workplace environment, this method is called whenever the user chooses the close button of a page.

You can deny closing a subsession in your implemented method:

```
public class ABCAdapter
    extends com.softwareag.cis.server.Adapter
{
    ...
    ...
    public void endProcess()
    {
        // veto the endProcess in case of unsaved data
        if (changedDataNotSaved == true)
        {
            this.outputMessage("E","Please save data first");
            return;
        }
        // close subsession
        super.endProcess();
    }
}
```

Call Sequence when a Session is Destroyed

If a session is removed from Application Designer - for example, if the user closes the browser or if a system administrator removes the session - the adapter instances are informed in the following way:

- **destroy()**
  In your implemented method, clean up all resources bound to your adapter instance. You cannot deny the destroying of the session - but you can react.

Error/ Runtime Exceptions

Error and runtime exceptions occurring during the adapter request processing may be handled centrally inside your adapter. For more details, see *Binding between Page and Adapter* in the *Special Development Topics*.

Pay Attention when Overwriting

The methods named above are already implemented with default behavior inside the class `com.softwareag.cis.server.Adapter`. Pay attention when overwriting these methods inside your adapter and always include the super-class's processing into your own implementation. The first statement inside your implementation should call the super-class method:

```
public class ABCAdapter
    extends com.softwareag.cis.server.Adapter
{
    ...
    ...
    public void reactOnDataTransferStart()
    {
        super.reactOnDataTransferStart();
        // now own implementation
        ...
        ...
    }
}
```