Session Management

You might ask: who controls the life cycle of the adapter classes? If I navigate from page "A" to page "B" and go back to page "A": do I come back to the adapter object I was already using, or do I get a new adapter instance?

This chapter covers the following topics:

- Session, Subsession, Adapter
- Garbage Collection

Session, Subsession, Adapter

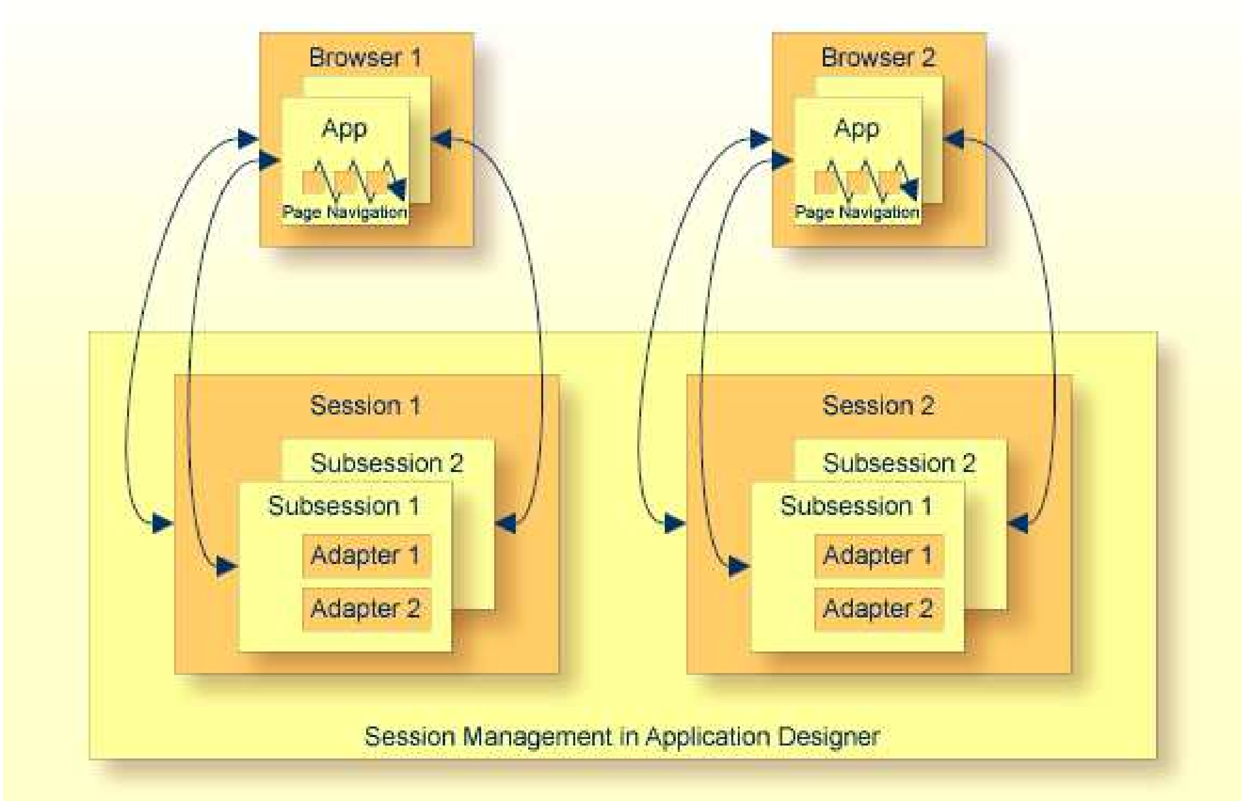
The management of the adapters inside the server is done by the session management of Application Designer. Typically you do not have to take care of it - it is done automatically in front of your adapters.

Every browser instance connected to Application Designer creates a session and is assigned to it at the server side. If you start another browser instance, a second session is created internally which is completely decoupled from all other sessions. And so on.

A session is divided into subsessions. A subsession is a logical separation of independent activities which run parallel within the context of one session. Example: in the workplace, you can run various applications in parallel. You can switch from one application to the other. Each running application is represented by an instance of a subsession at the server side. The subsessions are also completely isolated from each other.

Within a subsession, the adapter instances are held. The basic rules for managing these instances inside one subsession are:

- For each adapter class one instance is kept. This means: if a page requests an adapter, it is first determined whether this adapter instance is already created within the subsession. If yes, the existing instance is used, otherwise a new adapter instance is created and registered.
- The adapter instance is held for the whole life cycle of the subsession - as long as not explicitly removed by the adapter logic.
- All variant and page navigation is done inside a subsession as described in this section.



Page navigation within the browser is a navigation between adapter instances of the same subsession.

Garbage Collection

The final garbage collection of adapter instances is done by removing a subsession - if not explicitly controlled in a different way by the adapter logic. The adapter class offers the method `endProcess()` which removes the subsession you are just working with:

```
public void exit()
{
            // check if you really want to exit
            if ( ... )
            {
                ...
                this.outputMessage("E","Cannot exit due to...");
                return;
            }
            // exit
            this.endProcess();
}
```

Whenever a user logs off, the session - including all subsessions and its assigned adapter instances - is removed from the session management and released for garbage collection.