Background Information

This chapter covers the following topics:

- Link to Session Management
- Performance Considerations
- URL Position of the Pages
- Dynamic Pages - Multi Language Management

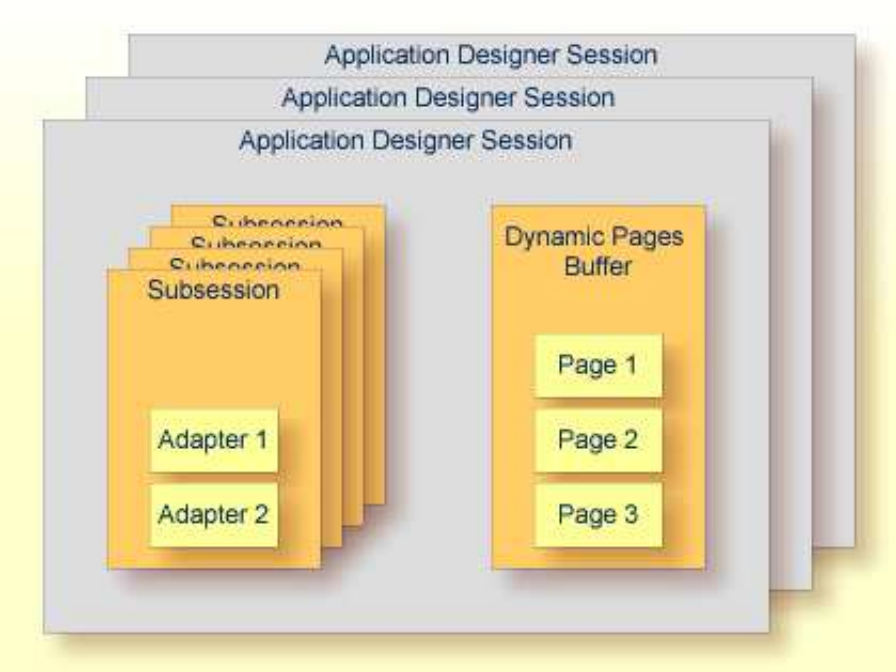
Link to Session Management

The dynamic page management is linked to the session management. Dynamic pages are generated from an XML definition. When calling the method `addDynamicPage(...)` from the interface `IDynamicPageMgmt`, an HTML string is internally generated and kept. It is later picked up by the servlet URL that references the dynamic page.

This means that two aspects are important:

- When is the page taken out of the memory?
- Who can use the page besides the one who has generated it?

Both questions are answered with reference to the session management. See also *Details on Session Management*.



The above diagram illustrates that a generated page belongs to one session. When the session is removed (e.g. due to log out of the user or due to timeouts), all dynamic pages are released to garbage collection. Of course, you can also remove dynamic pages by using the `removeDynamicPage(...)` method.

Only the session that has created the dynamic page can use it. Parallel sessions are not able to see it; they have to have their own dynamic pages, if required.

Performance Considerations

Dynamic pages are a very flexible technology for building generic application parts. However, this flexibility has some disadvantages when looking at the consumption of resources:

- Normal pages are generated during the design time process; they are already "compiled". Dynamic pages require an extra generation step during runtime before they can be used.
- Normal pages do not burden the memory because they are stored inside the file system. Dynamic pages are kept in memory. A large page with many controls can be in an area of more than 50-100 kBytes of HTML and JavaScript code. Keep in mind that every user who is logged on holds instances of the pages in the corresponding session context.

Therefore, you should only use dynamic pages when you have specific requirements.

URL Position of the Pages

Normal intelligent pages are located inside a project directory inside the web application that includes Application Designer. Internally, the page is addressed with the following URL:

```
http://<host>:<port>/<webapplication>/<project>/<pagename>.html
```

Note:
Remember that you normally do not directly reference pages because they always have to be embedded into a certain environment which is created by the StartCISPage servlet.

If a certain icon is addressed inside the page, the URL of the icon is typically relative to the page's position. Typically, images are kept in a separate directory below the project - e.g. an icon image is positioned inside an *images* directory. In this case, the image is addressed in the following way:

```
images/iconimage.gif
```

Dynamic pages are referenced by the internal usage of a special servlet. The URL that is internally used to access a dynamic page is:

```
http://<host>:<port>/<webapplication>/servlet/StartDynamicPage?SESSIONID=<sessionid>&DYNAMICPAGE=<pageid>
```

This means that from the URL reference point of view, your page is living below the URL root:

```
http://<host>:<port>/<webapplication>/servlet/
```

If you now reference resources which are inside your project's directory, you have to explicitly step into the project. The same icon that was used before, is now referenced via the link:

```
../<project>/iconimage.gif
```

Dynamic Pages - Multi Language Management

For the same reason as explained in the previous section, you must explicitly define the project in which the page is to live when using the multi language management. Multi language files are kept per project; consequently, a page needs to know the project from which it is to take the translated literals.

You define the project by using the following method of `IDynamicPageMgmt` and pass the name of the project.

```
public void addDynamicPage(String xml,
                           String name,
                           String applicationProject);
```

You can access the name of the project in which a concrete adapter is living by calling the Adapter method `findPageApplication()`.