JSF Portlet Development Tips and Suggestions

Version	Remarks	Released On
1.0	Initial Release	April 13, 2010

Question

For a JavaServer Faces (JSF) portlet application, higher memory consumption is reported. Is this an expected behavior or as an application developer, I am required to take care of certain aspects of JSF Portlet programming model ?

Cause

This section describes some of the topics related to the use of JSF and the capabilities provided with the IBM JSF Portlet Bridge that a developer should be aware of, to optimize the design of an application with regard to memory consumption vs. application behavior.

Note: In IBM Rational Application DeveloperV6.x, JSF Portlet Bridge is available under the lib folder of an application as a `jsf-portlet.jar`, and as a `jsf-portletbridge.jar` in IBM Rational Application Developer V7.x, and later.

1. Portlet Render Parameters: `com.ibm.faces.portlet.USE_RENDER_PARAMETERS`

This context parameter has been introduced for IBM WebSphere Portal V6.1 and is available in the portlet bridge (`jsf-portletbridge.jar`) from IBM Rational Application Developer V7.0.0.7, and later.

Description:
For servlet based Web application, the application server processes each browser request as single request. However, in a portal application the browser request is processed in two phases – an Action phase followed by a Render phase. Information traversing between Action to Render phase leaves a room for information loss, and this is a challenge for Web based framework such as JSF. The information that's configured in the JSF "request scope*"* gets destroyed during Action phase processing and isn't available for portlet rendering. The information loss is a deviation from the normal Web application paradigm, where information in "request scope*"* should persist until server generates a response.

This dilemma can be fixed in two ways, but each approach has its limitation:

- **Using Session**: Session used in Portlet Bridge in Rational Application Developer V6.0 (`jsf-portlet.jar`) may have reverse impact on the performance. If used extensively, the portlet applications will not be able to use the Back button and Refresh functionality of the browser.
- **Using Render Parameters**: This parameter is introduced in `jsf-portletbridge.jar`, that is, Rational Application Developer V7.0 to enhance the browser performance. Information passed from the Action phase to Render phase is kept in this render parameter during Action phase, making it available in the Render phase. This approach is useful when browser features such as Back button and Refresh is of importance for the application designers and doesn't interfere with session. However, this approach has its limitations:

- The render parameter can only store String objects.
- The object passed from Action phase to Render phase is encoded and decoded into Strings.
- The object is implemented in Serializable interface; without it encoding isn't possible.

More details on the relative merits and demerits can be had from this technote: JSF: Serializing and passing data from the Action phase to the Render phase for portlet bridge

How to decide on the choice of approach?

With IBM Rational Application Developer V7.0, render parameters are used by default. Thus, it's sole discretion of application designer to use it or not, by specifying the appropriate value in the application's web deployment descriptor.

One caveat to use Render parameters is that the String objects can be of considerable size (more than 2K) and in such a case Portlet Bridge will store the value in session and place corresponding key in the URLs generated into the page.

Tip: Avoid the expense of saving managed beans as Render Parameters. In the Action phase explicitly set Render Parameters (for example, a customer id) to be used in the rendering. Create a new "request scoped" managed bean in the Render phase. Property set into this new bean can be the value of a Render Parameter.

The portlet bridge (`jsf-portletbridge.jar`) supports both approaches, using Render Parameters is the default approach. To turn off the use of Render Parameter, set its value to `false` in applications' `web.xml`.

```
<context-param>
   <param-name>com.ibm.faces.portlet.USE_RENDER_PARAMETERS
   </param-name>
   <param-value>false</param-value>
</context-param>
```

2. Use of Multiple Action Execution (`wps.multiple.action.execution`)

Description:
This is a portlet initialization parameter that can be set in the portlet's deployment descriptor (that is, `portlet.xml`). This feature is used by IBM WebSphere Portal to stop processing the same Action request twice (for example, browser Back button feature). If this protection feature is left on (`wps.multiple.action.execution=true`), WebSphere Portal would treat a repeated Action URL as a Render URL with no repetition of portlet action, rather portlet would just be rendered. This is achieved by storing executed Action results and state in a session to prevent the multiple actions.

JSF portlets may utilize large render parameters that are stored in portlet action results, so using the multiple action protection features with this storage of action results can cause in substantial memory consumption.

Tip: To avoid high memory consumption, additional portal configuration settings that have multiple Action execution and result caching enabled can be used while capping the memory utilization. These can be used to specify a cache size boundary for Action ID keys and result state values. These are:

Setting: `wps.multiple.action.cache.bound.enabled`
Values: `true|false (default = false)`

Setting: `wps.multiple.action.cache.key.maxsize`
Values: `int >= 1 (default = 20) (Must be larger than wps.multiple.action.cache.value.maxsize)`

Setting: `wps.multiple.action.cache.value.maxsize`
Values: `int >= 1 (default = 5)`

A typical setting would be:

```
wps.multiple.action.cache.bound.enabled=true
wps.multiple.action.cache.key.maxsize=40
wps.multiple.action.cache.value.maxsize=10
```

- `wps.multiple.action.cache.key.maxsize` stores executed Action IDs of last 40 entries, thus preventing multiple execution (say when using Back button). This data is usually of few bytes, so setting a higher value doesn't utilize high memory.

- `wps.multiple.action.cache.value.maxsize` stores the Action result state of the last 10 entries and will correctly show the result of past executed Actions (say in case of Back button). It is usually a large data, and along with JSF Render Parameters it can grow to several KBs per entry, resulting in high memory consumption.

These settings are available in IBM WebSphere Portal V6.1.0.2 thru APAR PM08078 and can be applied using IBM WebSphere Application Server admin console:

WAS admin console → Resource Environment Providers → WP ConfigService

3. Key Manager (`keymanager.lru.size`)

Description:

This is a portal server side configuration parameter.

The browser history expiration limit can be controlled by setting the value of this property to an integer value in the `StateManagerService`. From IBM WebSphere Portal infocenter –

> *Use this parameter to specify the history expiration limit of portal pages visited by users. This determines how far backwards users can at least navigate in the recent history of portal pages that they visited. The number that you specify defines the minimum number of different pages selected by the user after which the portal can discard the render parameters of a page. (The decision whether the render parameters of the page are actually discarded depends on the expiration policy of the internal cache that stores the render parameters of those pages.) If the user returns to a page after visiting the specified number of other pages and if the render parameters of that page have expired, the portal displays that page in its default state.*

The description above indicates that this affects navigation between portal pages and not portlets.

Tip: If this parameter is enabled, number of Render Parameters in a session can somewhat be controlled. A typical setting would be

```
keymanager.lru.size = 3
```

4. Make use of Navigation Rules

In case rendering of same page is required, developers should follow the navigation rules. Same page rendering can happen in the following cases:

- If an Action returns `NULL` or an empty String, JSF will re-render the same page using the current JSF tree state.
- In some cases staying on the same page cannot be avoided, for example, a form is submitted and it has validation errors.

In portal due to the split of request into Action phase and Render phase, staying on the same page also needs to pass the tree state to the Render phase. Irrespective of approach used, it is an overhead for performance and memory usage.

Tip: If one's intent is to display the same page with different content, then a navigation rule should be set up to navigate to the page. In this case no state is sent to the Render phase.

Question

Are there any further tips that can be incorporated in general for JSF Portlets?

Answer

Yes, following tips would probably help -

1. *Use the most recent version of JSF Portlet Bridge*

   It's recommended to use the latest version of JSF Portlet bridge jar (`jsf-portletbridge.jar`). This jar is packaged and shipped with IBM Rational Application Developer. So, users are advised to update their IBM Rational Application Developer installations to the latest fix packs.

2. *Eliminate unnecessary and excessive per-user session memory usage*

   (For example, `<t:saveState>` from the Tomahawk library)
   Excessive use of the Tomahawk `<t:saveState>` tag increases the size of JSF view state information and should be removed from the code. The tag provides an easy-to-use capability that masks the complexity of per-user memory management (also known as session management) without solving the inherent problems of session management, such as:
   - storing redundant copies of application data within session memory thru t: saveState tags
   - failing to delete application data from session memory when appropriate

   Even a slight session management problems become significant causes of scalability and stability issues. It is suggested that the standard mechanisms available with J2EE and JSF should be used to enact the storage of temporary values.

3. *Keep page level backing beans in "request scope", if possible*
   Ineffective application designs lead to high cost of memory utilization and potential memory leak. The page level backing beans which are maintained in the "session scope" has high memory usage, and since backing bean is maintained in "session scope", it stays in all the screens at all the time until the user logout the application.

4. *Usage of parameter* `com.ibm.faces.portlet.UPDATE_LOCALE` *in web.xml*
   In Portal there is a button/link to select the language/locale to use. When this button is clicked, all the portlets are re-rendered. Since the portal has already rendered once, there exists a view state for the portlet page to render, so it will be "restored". If the bridge does not reset the locale, then the page would just re-render using the old locale. User can set the value of this parameter as true or false for updating the locale while restoring the view.

5. *In case of portlets that extend* `FacesPortlet` *and need to implement custom navigation between pages* (that is, anything that goes beyond the standard JSF navigation provided by a Navigation Handler), it's *recommended* to implement `getView()` method and return a valid View ID.

Question

How to determine the version number of IBM JSF Portlet Bridge.

Answer

IBM JSF Portlet bridge version is available in portlet bridge's jar at -

```
jsf-portletbridge.jar\META-INF\Manifest.mf
```

Appendix:

Recent fixes to IBM JSF Portlet Bridge (`jsf-portletbridge.jar`)

S.No	Complain	Fixed-in
1.	JSF graphic image tag was working on WP V6.0 but broken on WP V6.1 with WAS V7.0 `<hx:graphicImageEx styleClass="graphicImageEx" id="imasgeEx4" value="theme/icons/person.gif" hspace="5" border="0"></hx:graphicImageEx>`	RAD V7.5.5.1
2.	`portletPreferences` not working on WP V6.1.x on WAS V7.0	
3.	For JSR 286 portlets that implement new methods such as `serveResource` or `receiveMessage`, one was not able to acquire `FacesContext` easily . These new methods are not supported by the bridge and to implement these methods it's required to over-ride `getFacesContext` and `getLifecycle` methods. However, bridge provides private access to these modifiers. This has been relaxed to protected.	
4.	While migrating from WP V5.1 or WP V6.0 projects created using old RAD V7.0 versions to WP V6.1 the context-param `com.ibm.faces.portlet.USE_RENDER_PARAMETERS` in `web.xml` will be set to "`false`". This is to enable the bridge to use session instead of Render Parameters. This has now been automated, and RAD migration tool would set it during project migration.	
5.	A new parameter `com.ibm.faces.portlet.UPDATE_LOCALE` in `web.xml` has been added in RAD 7.5.3. User can set this parameter as true or false for updating the locale while restoring the view as explained in section 'Usage of parameter com… above'.	RAD V7.5.3
6.	Faces context and external context were not released because of which memory consumption was high. Context should be released after Action and Render phase.	RAD V7.5.3