Using Interportlet Communications

Sun Java™ System Portal Server 2005Q4 Technical Note

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 819–5942
February 2006

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

LA DOCUMENTATION EST FOURNIE "EN L'ETAT" ET TOUTES AUTRES CONDITIONS, DECLARATIONS ET GARANTIES EXPRESSES OU TACITES SONT FORMELLEMENT EXCLUES, DANS LA MESURE AUTORISEE PAR LA LOI APPLICABLE, Y COMPRIS NOTAMMENT TOUTE GARANTIE IMPLICITE RELATIVE A LA QUALITE MARCHANDE, A L'APTITUDE A UNE UTILISATION PARTICULIERE OU A L'ABSENCE DE CONTREFACON.

Inter Portlet Communication

This technical note contain the following sections:

- "Technical Note Revision History" on page 3
- "Introduction to Inter Portlet Communication" on page 3
- "Inter Portlet Communication API" on page 3
- "Developer Samples" on page 6
- "Accessing Sun Resources Online" on page 6

Technical Note Revision History

Version	Data	Description of Changes
10	02/2006	Final release of this technical note

Introduction to Inter Portlet Communication

The Portal Server software includes an application programming interface (API) which is an extension to JSR-168 Portlets, in the `com.sun.portal.portlet` Java package. Using this API, JSR-168 portlets will be able communicate with each other even if they are in different web applications. It is assumed that all these portlets will be on the same instance of a Portal Server and running inside the Portal Server Portlet container.

Inter Portlet Communication API

This section contains the following:

- "Overview of Inter Portlet Communication" on page 4
- "Event Generation and Subscription" on page 4
- "Event Handling Life Cycle" on page 4
- "Scope of Event Processing" on page 5
- "Infinite Event Cycle Detection" on page 5
- "Deterministic Behavior" on page 5

Overview of Inter Portlet Communication

This API uses event generation and notification to convey the information/data among portlets. The event notification will be for the portlets which have registered themselves for listening to that particular event. Portlets interested in receiving an event will implement a single interface `PortletEventListener`.

The `EventRequest` interface can then be used to obtain the event name and event payload data. Event payload data can be obtained either by getting the event stream and reading from it or by calling `getEventData()` method which returns an Object and then casting it appropriately.

The `EventResponse` interface can then be used to set the render parameters so that the information can be passed on to the render method after processing the event received in `handleEvent()` method. The portlets can generate events only from within the `handleEvent()` or `processAction()` methods. Event can be generated by instantiating `PortletEventBroker` and calling `createEvent()` method on it.

Event Generation and Subscription

Event generation starts with an event generated in `processAction()` method of a portlet. Further events can also be generated in `handleEvent()` method in portlet class while handling the event received. The `handleEvent()` method will be called if and only if an event is fired and the portlet has subscribed to listen to that event. The events can also be generated in response to other events in `handleEvent()` method.

To create portlets which are interested in listening to certain events and taking some action in response to these events, the portlet must implement `PortletEventListener` interface. This interface has only `handleEvent()` method. The portlet gets the data from *EventRequest* and can take appropriate action. The developer can set any information required for the correct rendering of the portlets on *EventResponse* as render parameters.

All the portlets which are interested in listening or generating an event must declare it in `sun-portlet.xml` file. If a portlet requests an event which has not been declared in the `sun-portlet.xml` file, an exception `NotRegisteredException` will be thrown. Wild cards can not be used for declaring the events that will be generated. However, portlets interested in consuming all the events can use wildcard character (*) only inside `<consumes–event></consumes–event>` block.

Event Handling Life Cycle

The event cycle begins with the response to user interaction from inside `processAction()` method referred to as Generation 1 events. These events are placed in the event queue by the Portlet Container and dispatched in the order they are created. The dispatching of the events continues till all the events in the event queue are dispatched to appropriate portlets. Dispatching of the events amount to calling the `handleEvent()` methods of the appropriate portlets. Portlets can generate events in the `handleEvent()` method which are referred to as Generation next events. If a portlet has subscribed to events which are generated in different generations, it will receive the events in proper order; that is, the `handleEvent()` method will be called with Generation i event first and upon completion of that method, `handleEvent()` method will be called with event from Generation `i+1`.

Scope of Event Processing

The events are sent to all the portlets obtained by recursively calling `getSelectedChannels()` method on the top level container (if any) as referenced by the request.

Infinite Event Cycle Detection

Events are generated in response to the user interaction (Generation 1) or in response to other events (Generation next). This could lead to more and more generations being created. To control the number of generations, the *maxEventGenerations* parameter in the `desktopConfig.properties` file can be configured for maximum number of generations of events per request. When the event creation exceeds the specified maximum number, a failure event, `eventHandlingFailed` will be sent to all the participating portlets.

Deterministic Behavior

If a portlet generates events X and Y in that order, then events X and Y will be delivered to the portlets in that same order. If portlet A and B are interested in Event X, either A or B can get event X first. If portlets A and B are interested in Event X, and upon receipt of that event generate events Y and Z respectively, and Portlet C is interested in Event Y and Z, then portlet C can receive events Y and Z in any order.

Failure and Exception Handling

In case of failure, the `handleEvent()` method may throw `PortletException`. The container catches that exception and will stop sending the events from the event queue. The container will then send another event called `eventHandlingFailed` to all the portlets participating in that particular interaction. The container will not take any action if the `PortletException` is thrown while processing `eventHandlingFailed` event. Portlets can not generate and send any events while handling the event `eventHandlingFailed`.

Developer Samples

Two developer samples are included. The first sample demonstrates first-order eventing to communicate information from source portlet to target portlet. In this case, the source portlet fires one event and the target portlet gets that event and updates its rendering.

In the second sample, a portlet generates an event which is delivered to the target portlet. In response to this event, the target portlet generates another event, which is consumed by a third portlet which updates its rendering in response to the event received.

Accessing Sun Resources Online

The docs.sun.com^{SM} web site enables you to access Sun technical documentation online. You can browse the docs.sun.com archive or search for a specific book title or subject. Books are available as online files in PDF and HTML formats. Both formats are readable by assistive technologies for users with disabilities.

To access the following Sun resources, go to `http://www.sun.com`:

- Downloads of Sun products
- Services and solutions
- Support (including patches and updates)
- Training
- Research
- Communities (for example, Sun Developer Network)

Third-Party Web Site References

Third-party URLs are referenced in this document and provide additional, related information.

Note – Sun is not responsible for the availability of third-party web sites mentioned in this document. Sun does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Sun will not be responsible or liable for any actual or alleged damage or loss caused or alleged to be caused by or in connection with use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions. To share your comments, go to `http://docs.sun.com` and click Send Comments. In the online form, provide the full document title and part number. The part number is a 7-digit or 9-digit number that can be found on the book's title page or in the document's URL. For example, the part number of this book is 819-5942-10.