intra-mart WebPlatform/AppFramework Ver.7.2

Portlet Programming Guide

First Edition: 2012/05/07

<< Revision History >>

Revised Date	Revision details
2012/05/07	First Edition

<< Table of Contents >>

1 Introduction

1.1 Purpose

This document explains how to create a portlet module which can be used by displaying in the intra-mart portal screen.

This document mainly deals with outline of portlet programming and notes on creating portlets.
Accordingly, sample codes described in this document are those excerpted partially.
In order to create portlet applications which actually can be operated, it is necessary to understand the language specifications of each development model and characteristics of the portlets, and then to implement such applications.

The intra-mart WebPlatform/AppFramework have some portlet samples included. Refer to them together with this document when creating a portlet application.

2 Outline

2.1 About intra-mart portal module

Portal modules operate as portlet containers stipulated in JST168 and JSR286.

So portlets complying with JST168 and JSR286 can be used without change.

- JSR168

  Standard specifications ver. 1.0 on portlets, formulated by a Java standardization body (Java Community Process)

  It includes definitions of features needed as portlets and portlet containers, and API for creating portlets (Portlet API 1.0).

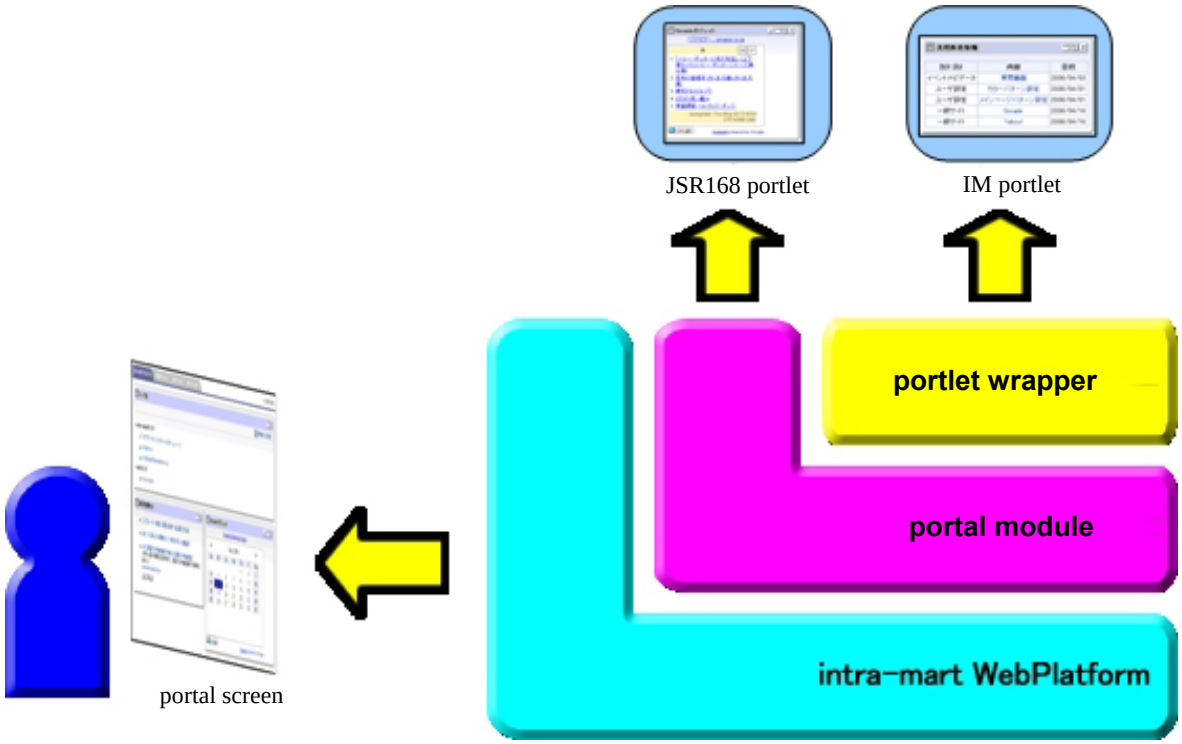
- JSR286

  Standard specifications ver. 2.0 on portlets, formulated by the Java standardization body (Java Community Process)

  It defines API (Portlet API 2.0) and specifications of additional features for JSR168. These are defined in a manner that maintains compatibilities with portlets complying with JSR168. 。

In addition, wrappers for using intra-mart applications as portlets have been provided by default, enabling portlets used with older versions to be used without change.

In this document, in order to distinguish portlets defined by JCP and those created by intra-mart applications, the former are called “Java portlets”.



< intra-mart portal module structure >

2.2 Types of portlet

Portlets which can be used for portal modules are largely categorized into two types as follows:

1. portlets created as intra-mart applications;
2. Java portlets.

In addition, the former is further categorized into three types as follows:

< Types of portlet >

	Types of portlet	Development model
1	portlets created as intra-mart applications	Script Development Model
2		JavaEE Development Model
3		JSP/Servlet
4	Java portlets	JavaEE Web application

In the next chapter, how to develop portlets are explained for each of these portlet types.

2.2.1 Portlet development using JSP/Servlet

Other than as mentioned above, it is possible to register Web pages developed with JSP and Servlet, not using intra-mart framework.
Such portlets can be created with standard development methods for Web applications.

However, some Web application frameworks like Struts are intended for creating ordinary Web pages and therefore are not suitable for creating portlets.
In most cases these cannot be used as portlets without change, and therefore it is necessary to modify them for using as portlets.
Specifically, please take note of the following points when modifying them:

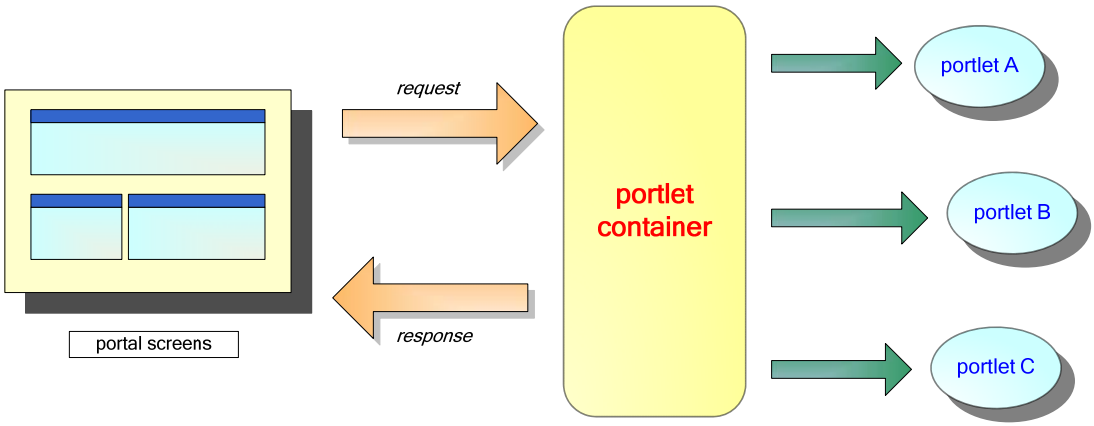
- Since portlets are called by INCLUDE processing from the portal screen, FORWARD processing cannot be handled during handling the portlet.
  In Struts, FORWARD is automatically handled unless its processing is specifically described, so make sure that RequestDispatcher#include() is called within the process of Action class.
- Elsewhere, it should be noted that FORWARD processing within struts-config.xml cannot be used.
- In addition, target of ServletFilter is only FORWARD, so if it is necessary to use Filter, add INCLUDE to the target.

2.3 Lifecycle of Java portlet

For Java portlets, the following lifecycle methods are defined in order that the portlets can be operated cooperatively with portal screens and with other portlets.

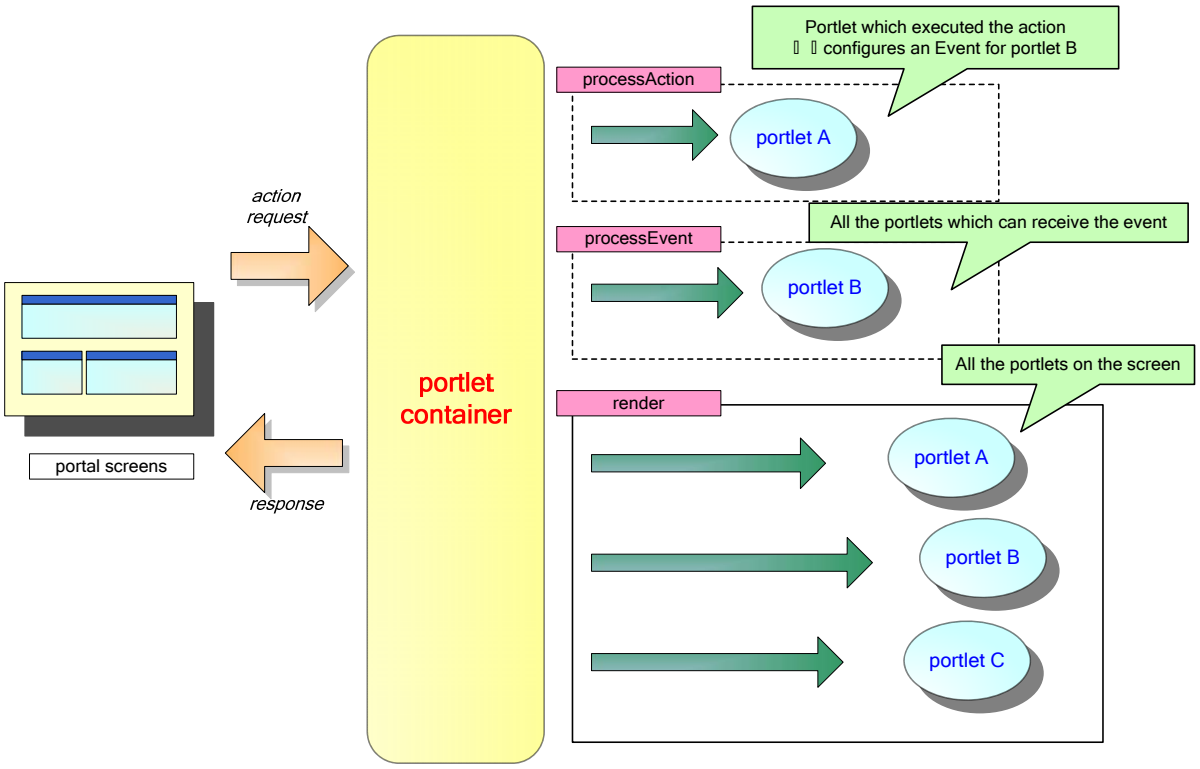
- processAction: method for executing processing for portlets
- processEvent: method for reception processing for events derived from processAction
- render: render processing of portlet screens

Usually, requests towards portal screens are accepted by a portlet container, which then calls render methods for each portlet.



Portal: render processing flow

In portlets, processAction can be called by submitting form data to the action URL by means of optional buttons and controls. In addition, by making an Event occur within processAction, processEvent of the relevant portlet can be called.



Portal: processAction/processEvent processing flow

In order to call processEvent, it needs to be configured for the portlet container in advance. For details on the configuration, refer to the chapters for each development model.

2.4 Portlet mode

There are several modes for portlets as follows, the displayed contents of which can be controlled by users switching them.

- VIEW mode
- EDIT mode

- HELP mode
- CONFIG mode: this is for configuration portal and is not displayed on regular portal screens.

However, in order to support these modes, it is necessary to configure the mode used for the portlet container in advance.
For how to configure, refer to the chapters for each development model.

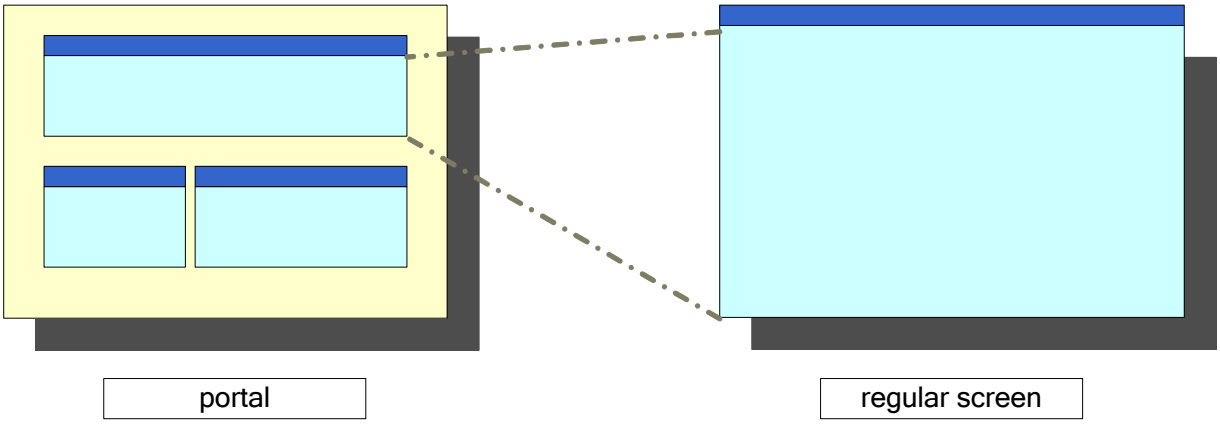
2.5 Window status

There are several window status for portlets as follows, the displayed contents of which can be controlled by users switching them.

- NORMAL
- MAXIMIZE
- MINIMIZE: the content will not be displayed, so the portlet will not be called.

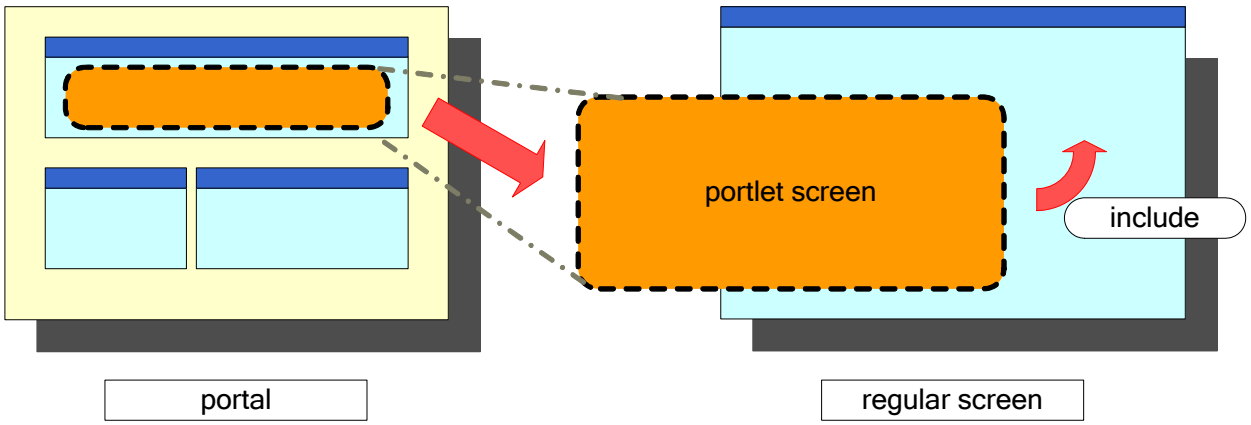
2.6 Screen for portlet

Portlet screen is displayed as a part of portal screen, so it cannot be created in the same way as regular screens. For details, refer to [7 Notes].



Comparison of portal screen and regular screen

For this reason, even if a common screen for regular screen and portlet screen should be used, separate screens need to be created. However, since screen display and features can be configured as common; so by creating a screen which would include portlet screen as described below, there would be no need to recreate another screen for regular screen.



Sharing regular screen and portlet screen

For how to implement them, examples are shown for each development model. Refer to sections [3.5.3] (Script Development Model) or [4.5.3] (JavaEE Development Model).

3 Portlet development (script development)

3.1 Outline

This chapter explains procedures of and notes on creating portlets using Script Development Model. While Script Development Model has several restrictions, features defined by JSR168 and JSR286 can be used.
In addition, in order to create portlets, it is necessary to understand the content of [2.3 Lifecycle of Java portlet] and determine which lifecycle should be used.

1. Screen development (render cycle)
   Screens displayed for portlets will be created. These can be switched by determining the mode and window status.
   Screen for view mode is always required.

2. Action processing (processAction cycle)
   Data processing submitted from portlets will be created. The function container created is called Action handler, which can be used through registration in the portlet editing screen.
   In addition, by configuring an Event, it becomes possible to make linkage with other portlet.

3. Event processing (processEvent cycle)
   Reception processing of an Event which occurred from other portlet will be created. The function container created is called Event handler, which can be used through registration in the portlet editing screen.
   This will be executed only when the Event is configured with Action handler.

For each portlet lifecycle, files to be created are different as follows:

- render: required; screen for Script Development Model; presentation page and function container.
- processAction: optional; function container implemented with handleAction function.
- processEvent: optional; function container implemented with handleAction function.

By creating programs for every lifecycle and combining them, such programs can be made as one portlet; and by developing separately for each lifecycle, they can be created so as to be used in combination among multiple portlets.

The next section and onward explain APIs available for portlets and how to implement them in each lifecycle.

3.2 Portlet API

Portlets consisting only of simple screen displays can be created without using portlet API, whereas in order to make maximum use of portlet features, it is necessary to use portlet API.
Especially, Action processing and Event processing cannot be executed without making use of portlet API.
In this document, explanations are limited to basic functions. For all of the APIs, refer to the separate document "Portal API List".

3.2.1 PortalManager

This provides various functions for using portlet features.
Explanations on each function are given in the next section and onward for their practical use.

3.3 Portlet mode

Page type of a portlet created by Script Development Model is "Presentation Page".
Whether to allow or deny using portlet mode is configured for each page type, and configured by default for the "Presentation Page" portlet as to deny using other than view mode.

By changing the system configuration, it becomes possible to use edit mode and help mode.
However, it should be noted that by doing such configuration, configurations for portlets with "Presentation Page" type will always be identical, thereby making all the modes usable.
In portlets with no edit mode implemented there would be no problem because the operation is same as view mode, but it is recommended that in the portlet editing screen the checkbox of whether to allow or deny use by users should be checked off as far as possible.

3.3.1 Portlet mode configuration

1. Changing portlet.xml
   - Portlet mode configuration is administered by the following file. Stop Application Runtime and configure the portlet mode, and then restart the portlet.

< Configuration file >

```
<%root of Application Runtime%>/doc/imart/WEB-INF/portlet.xml
```

< Configuration points >

```
        :
  <portlet>
        <description>Presentation Page Portlet</description>
        <portlet-name>PresentationPagePortlet</portlet-name>
        <portlet-class>jp.co.intra_mart.foundation.portal.portlets.model.PresentationPagePortlet</portlet-class>
        <expiration-cache>0</expiration-cache>
        <supports>
              <mime-type>text/html</mime-type>
              <portlet-mode>VIEW</portlet-mode>
```

```
            <portlet-mode>EDIT</portlet-mode> <!-- ← add edit mode -->
            <portlet-mode>HELP</portlet-mode> <!-- ← add help mode -->
        </supports>
        :
<portlet>
        :
```

2. Initializing IM portlet

- After configuring as described in subsection 1 above, log in as a system administrator and execute [Portlet administration] – [Initialize standard portlet] menu.

3.3.2 Getting portlet mode

By using API, current portlet mode can be gotten.

It should be noted that portlet mode is gotten in lower case.

- Ex) view, edit, help

```
var renderRequest = PortalManager. getRenderRequest();
var portletMode = renderRequest. getPortletMode();
```

3.4 Window status

3.4.1 Getting window status

By using API, current portlet mode can be gotten.

It should be noted that window status is gotten in lower case.

- Ex) normal, maximized, minimized

In addition, portlet will not be called while being minimized, so “minimized” status will not be gotten.

```
var renderRequest = PortalManager. getRenderRequest();
var windowState = renderRequest. getWindowState();
```

3.5 Screen development (render cycle)

In render cycle, screen display is carried out in the same way as regular Web pages.

Screen of Script Development Model is almost same as in regular screen development of intra-mart.

The page argument configured in the screens for new portlet registration/portlet editing (refer to “Portal Group Administrator Operation Guide”) can be gotten by the request object of init function.

In addition, the page argument configured by the Action processing and/or Event processing as described in the next section can be gotten as well by the request object of init function.

3.5.1 RenderRequest

Request object used by portlet is, unlike requests in regular Web pages, dedicated to portlet. By making use of this request, portlet information can be gotten.

- **Getting RenderRequest**

```
var renderRequest = PortalManager.getRenderRequest();
```

However, in order to share with regular Web pages, page argument can be gotten by using the request object of init function.

- **Getting the page argument**

```
function init(request) {
   var renderRequest = PortalManager.getRenderRequest();

   // value1 and value2 are identical
   var value1 = request.param1;
   var value2 = renderRequest.param1;
}
```

From the RenderRequest, current portlet mode and window status, etc. can be gotten.

3.5.1.1 Scope of RenderRequest

Scope of RenderRequest is different from the those of regular Web applications.

Every portlet retains independent request and does not inherit request parameter when displaying portal screen.

However, the following parameters will be configured automatically by portlet container:

- portal_cd: key for the portal screen in which the portlet is located
- portalKind: portal kind of the portal screen in which the portlet is located. The following values will be gotten:
  - user: user portal
  - group: group portal
  - global: global portal

In order to configure request parameters in a portlet, it is necessary to execute Action processing or Event processing.

In addition, the request parameter once configured will remain valid during the session until Action processing or Event processing is executed again.

3.5.2 ActionURL, RenderURL

When executing submit processing from the portlet created by Script Development Model and then creating a feature which displays portal screen again after finishing the processing, it is necessary to submit to the following URLs in order to communicate with the portlet container:

- ActionURL: URL for calling Action feature
- RenderURL: URL for displaying the portal screen again

These can be gotten by using PortalManager.

◆ **Getting ActionURL and RenderURL**

```
var actionURL = PortalManager.createActionURL();
var renderURL = PortalManager.createRenderURL();
```

Below are the samples of calling Action processing after submitting by the portlet developed by Script Development Model.

1. Function container

```
// ActionURL
var actionURL = "";

function init(request) {
	// Gets the ActionURL
	actionURL = PortalManager.createActionURL();
}
```

2. Presentation Page

```
<!-- Specify the ActionURL -->
<!-- Specify the window with target attribute -->
<form name="sampleActionForm" action="<IMART type="string" value=actionURL></IMART>"
	target="IM_MAIN" method="POST">

	<!-- Values used in the Action processing -->
	<INPUT type="text" name="param1" value="">

	<INPUT type="submit" value="execute">

</form>
```

- Display of portal screen is assigned with “portal-portal_display.service”, a service ID on JavaEE framework.
  For this reason, re-displaying portal screen by submitting to this service ID-URL is possible, but in doing so portal information cannot be inherited; therefore, it is recommended that the method as described above should be used.

- There is no way to call Action processing other than submitting to the ActionURL.
  Register the Action handler in the portlet administration screen, and submit it to the ActionURL from the presentation page.
  If Action handler has not been defined, portal screen will simply be displayed again.

- Since ActionURL contains portal information, its data amount will increase accordingly. POST should be specified as the method.

3.5.3 To share regular screen and portlet screen

Create a code to be shared and a code which only requires regular screen as separate presentation pages, and in the presentation page for regular screen read the shared page by using include tag.
However, request parameter will not be inherited by using include tag; therefore it is necessary to define other variables which should be inherited individually.
In addition, when doing such use, it should be noted that each API of the portlet cannot be used.

< Presentation Page of regular screen to be shared with portlet screen >

```
<html>
<head>
  <link src="sample.css" type="text/css">

</head>
<body>
  <div>
    Header display information
  </div>

  <!-- Include shared code -->
  <!-- param1and param2 are the request parameter to be inherited -->
  <IMART type="include" page="sample/sample-portlet"
    param1=value1
    param2=value2
    />

  <div>
    Footer display information
  </div>
</body>
</html>
```

3.6 Action processing (processAction cycle)

Since there is no screen display in processAction cycle, only function container will be created.

Moreover, Action processing will be executed by registering the Action handler created in the portlet administration screen and submitting it to the ActionURL.

RenderParamter configuration, PortletPreference configuration and update processing of such as event configuration are handled by Action processing.

3.6.1 Creating Action handler

An example of Action handler which configures events is shown below.

1. Create a function container and define the following functions.
   (This function container is called Action handler.)

(ex.) sample/portal/sample_action.js

```
// Define the function named “handleAction”.
function handleAction(request, response) {

   // Get the request argument.
   var value = request.param1;

   // Configure the event (key: "event1"; value: value).
   response.setEvent("im_event1", value);

}
```

- Action handler may be described in the function container to be used by render, or Action handler class created with Java may be used.

An event can be made occur also by returning the following values, for example:

- ◆ Object: event having a pair of key and value of the property
- ◆ Others: event determined as character string, and key and value of which is identical
  However, in order to make an event occur for an IM portlet, a key beginning with “im_” needs to be specified.

- For the value configured by response.setEvent(), refer to the description of ActionResponse in “Portal API List”.

2. Register the Action handler to the portlet.

< New portlet registration >

3. Locate the screen created as described in [3.5 Screen development (render cycle)], and call the Action processing from the portlet.
After executing the Action handler, the screen will be displayed again.

< Portal screen – Action processing >

3.6.2 ActionRequest, ActionResponse

Arguments of Action handler, i.e. request and response are an ActionRequest object and ActionResponse object, respectively.

By using these objects, it is possible to get a request argument and to configure an event.
For details, refer to “Portal API List”.

3.6.3 RenderParameter configuration

The request parameter which can be gotten when displaying the screen (RenderParameter) is configured in Action processing or Event processing.
The request parameter at the time of calling Action processing is contained in ActionRequest, but it will not be inherited to render cycle, and therefore it needs to be explicitly configured if necessary.
In addition, all the RenderParameters configured in the past will have been cleared when calling Action processing.

(ex.) Configuring the RenderParameter in processAction cycle

```
// When configuring RenderParameter:
response.setRenderParameter("param1", "value1");

// When deleting RenderParameter:
response.setRenderParameter("param2", null);
```

(ex.) Using the RenderParameter in render cycle

```
String value = request.param1;
```

3.6.4 Event configuration

By configuring an event, it is possible to pass over its handling to multiple portlets other than the one which accepted the Action processing, through notifying the occurrence of the event.

(ex.) Configuring an event in processAction cycle

```
response.setEvent(eventId, eventValue);
```

3.6.5 Available Action handler

In intra-mart WebPlatform/AppFramework, the following Action handlers have been made available by default.
These are the classes implemented in Java language, but can be used for script development.

3.6.5.1 jp.co.intra_mart.foundation.portal.common.handler.DefaultPortletHandler

If, in the screens for new portlet registration/portlet editing, checkbox of Action handler is checked, this Action handler will be configured as the default value.
This Action handler gets a value from the request parameter and make an event occur using its key and value.
When preferring to make an event simply from the screen, this handler can be used without creating an Action handler.

In order to make an event occur, configure the following request parameter.

- **“portlet.event.” + arbitrary character string**

This creates the following event.

Property name	Description
id	arbitrary character string of the request parameter key
value	value of the request parameter

By configuring multiple request parameters, it is also possible to make multiple events occur.

3.6.5.2 jp.co.intra_mart.foundation.portal.common.handler.SetRenderParameterHandler

Since the request parameter when displaying portal screen has different scope from the one of portlet, it cannot be gotten from the processing within the portlet.

By configuring this Action handler, it becomes possible to use for portlets the request parameter when displaying portal screen.

3.7 Event processing (processEvent cycle)

Since there is no screen display in processEvent cycle, only function container will be created.
Moreover, Event processing will be executed by registering the Event handler created in the portlet administration screen and configuring the event through Action processing of any portlet.

RenderParamter configuration, PortletPreference configuration and update processing of such as event configuration are handled by Event processing.

3.7.1 Creating Event handler

An example of Event handler which configures RenderParameter is shown below.

1. Create a function container and define the following functions.
(This function container is called Event handler.)

(ex.) sample/portal/sample_event.js

```
// Define the function named “handleEvent”.
function handleEvent(event, request, response) {

   // Get event information.
   var eventId = event.id;

   var value = event.value;

   // Configure the request parameter used for screen display.
   response.setRenderParameter(eventId, value);

}
```

- Event handler may be described in the function container to be used by render, or Event handler class created with Java may be used.

2. Register the Event handler to the portlet.

< New portlet registration >

3. Locate the screen created as described in [3.5 Screen development (render cycle)] and the portlet which can call the event created as described in [3.6 Action processing (processAction cycle)] in the portal screen, and call the Action processing from the portlet for configuring the event.
Event handler will be called from the Action processing, and then the screen will be displayed again.

< Portal screen – Event processing >

3.7.2 EventRequest, EventResponse

Arguments of Event handler, i.e. request and response are an EventRequest object and EventResponse object, respectively.

By using these objects, it is possible to get a request argument and to configure an event.
For details, refer to “Portal API List”.

3.7.3 Event object

The first argument of Event handler is the event information object configured by the Action processing. It contains the following information:

Property name	Type	Value
id	String	Event ID configured by the Action processing. If omitted, character string “ImEvent” will be configured.
value	String	Event value configured by the Action processing. This will be gotten as character string even when configuring an object.
source	String	Portlet code of the portlet which configured the Event.

3.7.4 RenderParameter configuration

The request parameter which can be gotten when displaying the screen (RenderParameter) is configured in Action processing or Event processing.
The RenderParameter which had already been configured at the time of calling the Event processing has been contained in EventRequest, and will be automatically inherited to render cycle unless specifically handled. If inheritance is not required, it needs to be explicitly deleted.

- In the specification of JSR286, RenderParameter will not be automatically inherited. It should be noted that this is a specification unique to the intra-mart portlet containers.

(ex.) Configuring the RenderParameter in processEvent cycle

```
// When configuring RenderParameter:
response.setRenderParameter("param1", "value1");

// When deleting RenderParameter:
response.setRenderParameter("param2", null);
```

(ex.) Using the RenderParameter in render cycle

```
request.getParamter(eventId);
```

3.7.5 Event configuration

By configuring an event, it is possible to pass over its handling to multiple portlets other than the one which accepted the Event processing, through notifying the occurrence of the event.
However, doing event processing further in the event processing would result in lowering the performance, as well as making event loop occur. So pay sufficient attention when implementing.

(ex.) Configuring an event in processEvent cycle

```
response.setEvent(eventId, eventValue);
```

3.8 General-purpose New-Arrived-Portlet

In intra-mart portal module, portlet for displaying general-purpose newly arriving information has been made available. This general-purpose New-Arrived-Portlet displays the following items as newly arriving information:

- Category for indicating the type of newly arriving information
- Content of the newly arriving information (outline)
- Arrived date
- Information about the transition destination tied to the newly arriving information (link)

While general-purpose New-Arrived-Portlet has been created by using JavaEE Development Model, an interface is provided so that is can be used for Script Development Model.
In order to display unique newly arriving information in the general-purpose New-Arrived-Portlet, the portlet should be registered according to the procedure as described below.

1. Implementing a provider for getting newly arriving information.

Create a function container implemented with the following functions for getting information arrays to be displayed in the general-purpose New-Arrived-Portlet:

- Function name: getNewArrivedList
- Parameters:

Parameter name	JavaScript type	Description
user	String	user ID
group	String	login group ID
properties	Object	default parameter defined in the configuration file

This function needs to return the arrays of the following objects of newly arriving information:

- Newly arriving information object

Properties	JavaScript type	Description
category	String	Category for indicating the type of newly arriving information
contents	String	Content of the newly arriving information (outline)
url	String	URL for transition to the details on newly arriving information (link)
popup	Boolean	Flag for determining whether to display popup screen when transiting to the detailed information
date	String	Arrived date of the newly arriving information

<sample_provider.js>

```
// Implement getNewArrivedList function.
function getNewArrivedList(user, group, properties) {
```

```
  // Create the array of newly arriving information.
var values = new Array();

  for (var i = 0; i < result.countRow; i++) {
      values[i] = new Object();
      values[i].category = result.data[i].category;   // Category for indicating the type of newly arriving information
      values[i].contents = result.data[i].contents;   // Content of the newly arriving information (outline)
      values[i].date = result.data[i].arrived_date;    // Arrived date
      values[i].url = result.data[i].url;                  // Information about the transition destination tied to the newly arriving information (link)
      values[i].popup = ("1" == result.data[i].popup);         // Whether to call popup screen when referring to the link.
  }

  // Return the array of newly arriving information.
  return values;
}
```

2. Defining the implementation of the provider.

Information about the provider class implemented will be defined by creating the configuration file in XML format in an arbitrary location under Storage Service. Define the following information in the configuration file:

< Provider configuration file >

Element	Explanation
new-arrived-portlet/sort	Defines whether the newly arriving information which has been gotten should be sorted by category.
new-arrived-portlet/provider/provider-class	Implementation class of the provider. This is fixed.
new-arrived-portlet/provider/init-param	Defines the function container for getting newly arriving information in “pagePath” parameter. By configuring other arbitrary parameter, it can be used in the function container.

For example, when getting newly arriving information by using a provider for getting newly arriving information from the abovementioned Script Development Model program, procedure to define is as follows:

<sample_portlet.xml>

```
<new-arrived-portlet>
  <sort>true</sort>
  <provider>
    <!—Provider class for Script Development Model. This should be fixed. -->
    <provider-class>
      jp.co.intra_mart.foundation.portal.general_purpose_portlet.provider.PageBaseCallProvider
    </provider-class>
    <init-param>
      <!—Define the provider. -->
      <param-name>pagePath</param-name>
      <param-value>/hoge/sample_provider.js</param-value>
    </init-param>
  </provider>
</new-arrived-portlet>
```

By registering more than one provider, it is also possible to display multiple newly arriving information in one portlet.

For the detailed definition of configuration file, refer to “intra-mart WebPlatform/AppFramework Portal Configuration Guide”.

3. Registering the portlet.

Register the portlet for displaying newly arriving information from the group administrator's [Portal] – [Portlet] editing screen.
When configuring general-purpose New-Arrived-Portlet, it is necessary to configure "Page type", "Screen path or URL type" and "Page argument". For example, configuration in the sample is as follows:

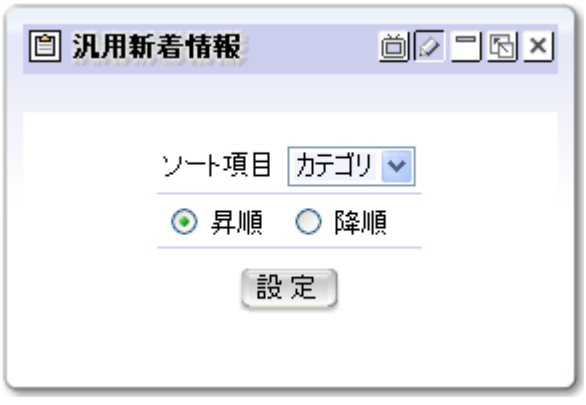
- Page type: Servlet/JSP
- Screen path or URL type: /portal/portlets/newly_arrived_view.jsp
- Page argument: page_param=/portal/sample_portlet.xml
- Action handler (option):
  jp.co.intra_mart.foundation.portal.general_purpose_portlet.handler.NewArrivedPortletHandler

The JSP to be specified as "Screen path or URL type" should be the one such as for displaying the newly arriving information which the provider has gotten. The JSP specified in the sample is a default implementation provided by the portal module.
For "Page argument", path of the configuration file which defined the provider information should be specified in "page_param= ..." format. For the value of page_param, path under the root directory of Storage Service should be specified.

3.8.1 Using edit mode

Page type of the general-purpose New-Arrived-Portlet is "JSP/Servlet", and edit mode is enabled by default.
However, in order to actually use edit mode, it is necessary to configure the Action handler as specified in the previous section, when registering the portlet.

When switching to edit mode, a screen as shown below will be displayed.
It is a feature only to switch between sort items and between ascending/descending order of the view mode. Usually a feature like this can be realized solely by the view mode, but this screen is incorporated with an implication as a sample of edit mode.
JSP/Java is used for implementation and therefore certain knowledge of Java is required, but equivalent processing can be executed by using Script Development Model.



< Edit mode of general-purpose New-Arrived-Portlet >

This feature consists mainly of the following modules:

- Mode switching module: As there is just one page path which can be registered, processing to determine the mode and to switch is required.
- Displaying module for view mode: This handles screen display in view mode.
- Displaying module for edit mode: This handles screen display in edit mode.
- Action handler for configuration: This handles processing when the [Configuration] button is clicked. As well as data registration, this handles mode switching after registration.

3.8.2 “Important notice” portlet

“Important notice” portlet is a uniquely created portlet application supporting JSR286, and its screen display processing is implemented by using the mechanism of general-purpose New-Arrived-Portlet.
Currently the following provider classes created by JavaEE Development Model are used, and information on notice from the system administrator and on password history administration will be displayed.

- jp.co.intra_mart.foundation.security.password.PasswordHistoryNewArrivedProvider
- jp.co.intra_mart.foundation.portal.portlets.system_notice.SystemNoticeProvider

Information to be displayed in the important notice is defined by the following file.

- <root directory of Storage Service>/portal/system_notice.xml

Providers of JavaEE Development Model and of Script Development Model can be present together, so by implementing the provider class with Script Development Model and adding definition to the configuration file, it becomes possible to deal with information other than as described above, as “important notice”.

< Important notice portlet definition file: system_notice.xml >

```
<new-arrived-portlet>
  <sort>true</sort>
  <provider>
    <provider-class>
      jp.co.intra_mart.foundation.security.password.PasswordHistoryNewArrivedProvider
    </provider-class>
  </provider>
  <provider>
    <provider-class>
      jp.co.intra_mart.foundation.portal.portlets.system_notice.SystemNoticeProvider
    </provider-class>
  </provider>
</new-arrived-portlet>
```

4 Portlet development (JavaEE development)

4.1 Outline

This chapter explains procedures of and notes on creating portlets using JavaEE Development Model. While JavaEE Development Model has several restrictions, features defined by JSR168 and JSR286 can be used.
In addition, in order to create portlets, it is necessary to understand the content of [2.3 Lifecycle of Java portlet] and determine which lifecycle should be used.

1. Screen development (render cycle)
   Screens displayed for portlets will be created. These can be switched by determining the mode and window status.
   Screen for view mode is always required.

2. Action processing (processAction cycle)
   Data processing submitted from portlets will be created. The function container created is called Action handler, which can be used through registration in the portlet editing screen.
   In addition, by configuring an Event, it becomes possible to make linkage with other portlet.

3. Event processing (processEvent cycle)
   Reception processing of an Event which occurred from other portlet will be created. The function container created is called Event handler, which can be used through registration in the portlet editing screen.
   This will be executed only when the Event is configured with Action handler.

For each portlet lifecycle, files to be created are different as follows:

- render: required; screen for Java EE Development Model; Main JSP and HelperBean.
- processAction: optional; Java class implemented with PortletActionHandler interface.
- processEvent: optional; Java class implemented with PortletEventHandler interface.

By creating programs for every lifecycle and combining them, such programs can be made as one portlet; and by developing separately for each lifecycle, they can be created so as to be used in combination among multiple portlets.

The next section and onward explain APIs available for portlets and how to implement them in each lifecycle.

4.2 Portlet API

Portlets consisting only of simple screen displays can be created without using portlet API, whereas in order to make maximum use of portlet features, it is necessary to use portlet API.
Especially, Action processing and Event processing cannot be executed without making use of portlet API.
In this document, explanations are limited to basic functions. For all of the APIs, refer to the separate documents “Portal API List” and “Portlet API 2.0 Document”.

- “Portlet API 2.0 Document” can be obtained from the following URL:
  http://jcp.org/en/jsr/detail?id=286

4.2.1 PortalManager

In JavaEE Development Model, PortletAPI 2.0 can be used without change, meanwhile PortalManager class has

been made available as an accessor for making easy use of these API.

```
jp.co.intra_mart.foundation.portal.common.PortalManager
```

Explanations on each function of PortalManager are given in the next section and onward for their practical use.

4.3 Portlet mode

Page type of a portlet created by JavaEE Development Model is “im-JavaEE Service Framework”.
Whether to allow or deny using portlet mode is configured for each page type, and configured by default for the “im-JavaEE Service Framework” portlet as to deny using other than view mode.

By changing the system configuration, it becomes possible to use edit mode and help mode.
However, it should be noted that by doing such configuration, configurations for portlets with “im-JavaEE Service Framework” type will always be identical, thereby making all the modes usable.
In portlets with no edit mode implemented there would be no problem because the operation is same as view mode, but it is recommended that in the portlet editing screen the checkbox of whether to allow or deny use by users should be checked off as far as possible.

4.3.1 Portlet mode configuration

1. Changing portlet.xml
   - Portlet mode configuration is administered by the following file. Stop Application Runtime and configure the portlet mode, and then restart the portlet.

< Configuration file >

```
<%root of Application Runtime%>/doc/imart/WEB-INF/portlet.xml
```

< Configuration points >

```
        :
  <portlet>
        <description>JavaEE Development Model portlet</description>
        <portlet-name> JavaeeFwPortlet </portlet-name>
        <portlet-class>jp.co.intra_mart.foundation.portal.portlets.model. JavaeeFwPortlet </portlet-class>
        <expiration-cache>0</expiration-cache>
        <supports>
              <mime-type>text/html</mime-type>
              <portlet-mode>VIEW</portlet-mode>
              <portlet-mode>EDIT</portlet-mode> <!-- ← add edit mode -->
              <portlet-mode>HELP</portlet-mode> <!-- ← add help mode -->
        </supports>
        :
  <portlet>
        :
```

2. Initializing IM portlet

- After configuring as described in subsection 1 above, log in as a system administrator and execute [Portlet administration] – [Initialize standard portlet] menu.

4.3.2 Getting portlet mode

By using API, current portlet mode can be gotten.

For portlet mode, constants of “javax.portlet.PortletMode” class will be gotten as listed below for example:

- Ex) PortletMode.VIEW, PortletMode.EDIT, PortletMode.HELP

```
javax.portlet.RenderRequest renderRequest = PortalManager. getRenderRequest();
javax.portletPortletMode portletMode = renderRequest. getPortletMode();
```

4.4 Window status

4.4.1 Getting window status

By using API, current portlet mode can be gotten.

For window status, constants of “javax.portlet.WindowState” class will be gotten as listed below for example:

- Ex) WindowState.NORMAL, WindowState.MAXIMIZED, WindowState.MINIMIZED

In addition, portlet will not be called while being minimized, so “WindowState.MINIMIZED” will not be gotten.

```
javax.portlet.RenderRequest renderRequest = PortalManager. getRenderRequest();
javax.portlet.WindowState windowState = renderRequest. getWindowState();
```

4.5 Screen development (render cycle)

In render cycle, screen display is carried out in the same way as regular Web pages.

Screen of JavaEE Development Model is almost same as in regular screen development of intra-mart.

The page argument configured in the screens for new portlet registration/portlet editing (refer to “Portal Group Administrator Operation Guide”) can be gotten by the getParameter method of HttpServletRequest object.

In addition, the request parameter configured by the Action processing and/or Event processing as described in the next section can be gotten as well by the HttpServletRequest object.

4.5.1 RenderRequest, RenderResponse

Request object and response object used by portlet are, unlike requests in regular Web pages, dedicated objects for portlet display, i.e. RenderRequest object and RenderResponse object respectively. By making use of these objects, portlet information can be gotten.

- **Getting RenderRequest object and RenderResponse object**

```
RenderRequest renderRequest = PortalManager.getRenderRequest();
RenderResponse renderResponse = PortalManager.getRenderResponse();
```

However, in order to share with regular Web pages, page argument can also be gotten by using HttpServletRequest (accessed in JSP with the keyword “request”).

- **Getting the page argument in JSP**

```
<%
RenderRequest renderRequest = PortalManager.getRenderRequest();

// value1 and value2 are identical
String value1 = request.getParameter("param1");
String value2 = renderRequest.getParameter("param1");

%>
```

From the RenderRequest, current portlet mode and window status, etc. can be gotten.

4.5.1.1 Scope of RenderRequest

Scope of RenderRequest is different from the those of regular Web applications.

Every portlet retains independent request and does not inherit request parameter when displaying portal screen.

However, the following parameters will be configured automatically by portlet container:

- portal_cd: key for the portal screen in which the portlet is located
- portalKind: portal kind of the portal screen in which the portlet is located. The following values will be gotten:
  - user: user portal
  - group: group portal
  - global: global portal

In order to configure request parameters in a portlet, it is necessary to execute Action processing or Event processing.

In addition, the request parameter once configured will remain valid during the session until Action processing or Event processing is executed again.

4.5.2 ActionURL, RenderURL

When executing submit processing from the portlet created by JavaEE Development Model and then creating a feature which displays portal screen again after finishing the processing, it is necessary to submit to the following URLs in order to communicate with the portlet container:

- ActionURL: URL for calling Action feature
- RenderURL: URL for displaying the portal screen again

These can be gotten by using PortalManager.

◆ **Getting ActionURL and RenderURL**

```
PortletURL actionURL = PortalManager.createActionURL();
PortletURL renderURL = PortalManager.createRenderURL();
```

Below is a sample of calling Action processing after submitting by the portlet developed with JavaEE development model.

1. JSP

```
<%
   PortletURL actionURL = PortalManager.createActionURL();
%>

<!-- Call a service to handle some registration processing -->
<!-- Specify the window with target attribute -->
<form action="<%= actionURL.toString() %>" target="IM_MAIN" method="POST">

   <!-- Values used in the registration processing -->
   <input type="text" name="param1" value="">

   <input type="submit" value="execute">
</form>
```

- Display of portal screen is assigned with “portal-portal_display.service”, a service ID on JavaEE framework.
  For this reason, re-displaying portal screen by submitting to this service ID-URL is possible, but in doing so portal information cannot be inherited; therefore, it is recommended that the method as described above should be used.

- There is no way to call Action processing other than submitting to the ActionURL.
  Register the Action handler in the portlet administration screen, and submit it to the ActionURL from the presentation page.
  If Action handler has not been defined, portal screen will simply be displayed again.

- Since ActionURL contains portal information, its data amount will increase accordingly. POST should be specified as the method.

4.5.3 To share regular screen and portlet screen

Create a code to be shared and a code which only requires regular screen as separate JSP pages, and in the JSP page for regular screen read the shared page by using include tag.

However, when doing such use, it should be noted that each API of the portlet cannot be used.

< Presentation Page of regular screen to be shared with portlet screen >

```
<html>
<head>
   <link src="sample.css" type="text/css">

</head>
<body>
   <div>
      Header display information
   </div>

   <!-- Include shared code -->
   <jsp:include page="sample/sample-portlet.jsp" flush="true"/>

   <div>
      Footer display information
   </div>
</body>
</html>
```

4.6 Action processing (processAction cycle)

Since there is no screen display in processAction cycle, only an Action handler class implemented with PortletActionHandler interface will be created.
Moreover, Action processing will be executed by registering the Action handler created in the portlet administration screen and submitting it to the ActionURL.

RenderParamter configuration, PortletPreference configuration and update processing of such as event configuration are handled by Action processing.

4.6.1 Creating Action handler

An example of Action handler which configures events is shown below.

1. Create a Java class implemented with PortletActionHandler interface, and define the following function. (This Java class is called Action handler.)

(ex.) sample.portal.SampleActionHandler.java

```
// Implement "PortletActionHandler" interface.
public class SampleActionHandler implements PortletActionHandler {

   public Serializable handleAction(String portletCd, ActionRequest request, ActionResponse response) {

      // Get the request argument.
      String value = request.getParameter("param1");

      // Configure the event.
      // When transmitting the event to a portlet for intramart, use ImEvent object.
      // Use ImEvent.IM_QNAME as QName.
      ImEvent event = new ImEvent();
      event.setEvent(value);
      response.setEvent(ImEvent.IM_QNAME, event);

      return null;

   }
}
```

- For Action handler, a function container that has been created with Script Development Model used for render may be used.
- An event can be made occur also by returning the following values, for example:
  - Instance of "jp.co.intra_mart.foundation.portal.common.handler.ImEvent"
  - Instance of "java.util.Map": event with a pair of key and value of the Map
  - Others: event determined as character string, and key and value of which is identical

  However, in order to make an event occur for an IM portlet, ImEvent class needs to be used, or a key beginning with "im_" needs to be specified.
- For details, refer to the description of PortletActionHandler in "Portal API List".

2. Register the Action handler to the portlet.

< New portlet registration >

3. Locate the screen created as described in [4.5エラー! ブックマーク名が指定されていません。 Screen development (render cycle)エラー! ブックマーク名が指定されていません。], and call the Action processing from the portlet.
After executing the Action processing, the screen will be displayed again.

< Portal screen – Action processing >

4.6.2 ActionRequest, ActionResponse

Arguments of Action handler are the dedicated objects for Action processing, i.e. ActionRequest object and ActionResponse object.

By using these objects, it is possible to get a request argument and to configure an event.

For details, refer to “Portlet API 2.0 Document”.

4.6.3 RenderParameter configuration

The request parameter which can be gotten when displaying the screen (RenderParameter) is configured in Action processing or Event processing.

The request parameter at the time of calling Action processing is contained in ActionRequest, but it will not be inherited to render cycle, and therefore it needs to be explicitly configured if necessary.

In addition, all the RenderParameters configured in the past will have been cleared when calling Action processing.

(ex.) Configuring the RenderParameter in processAction cycle

```
// When configuring the request parameter:
response.setRenderParameter("param1", "value1");

// When deleting the request parameter:
response.setRenderParameter("param2", null);
```

(ex.) Using the RenderParameter in render cycle

```
String value = request.getParamter("param1");
```

4.6.4 PortletPreferences configuration

PortletPreferences means portlet information storage area for each user.

In the portlet container of intra-mart, information configured in the PortletPreferences will be stored in the database.

Information configured here can be referred at any time while the screen is displayed.

(ex.) Using javax.portlet.PortletPreferences

```
// Get the PortletPreferences.
PortletPreferences preferences = request.getPreferences();

// Get the information configured in the PortletPreferences.
String value = preferences.getValue("key1", "Default Value");

// Configure the information in the PortletPreferences.
preferences.setValue("key1", "value1");

// Determinate the PortletPreferences.
// Unless this processing is executed, information will not be stored.
preferences.store();
```

4.6.5 Event configuration

By configuring an event, it is possible to pass over its handling to multiple portlets other than the one which

accepted the Action processing, through notifying the occurrence of the event.

(ex.) Configuring an event in processEvent cycle

```
response.setEvent(eventId, eventValue);
```

4.6.6 Available Action handler

In intra-mart WebPlatform/AppFramework, the following Action handlers have been made available by default.

4.6.6.1 jp.co.intra_mart.foundation.portal.common.handler.DefaultPortletHandler

If, in the screens for new portlet registration/portlet editing, checkbox of Action handler is checked, this Action handler will be configured as the default value.
This Action handler gets a value from the request parameter and make an event occur using its key and value.
When preferring to make an event simply from the screen, this handler can be used without creating an Action handler.

In order to make an event occur, configure the following request parameter.

- **"portlet.event." + arbitrary character string**

This creates the following event.

Property name	Description
id	arbitrary character string of the request parameter key
value	value of the request parameter

By configuring multiple request parameters, it is also possible to make multiple events occur.

4.6.6.2 jp.co.intra_mart.foundation.portal.common.handler.SetRenderParameterHandler

Since the request parameter when displaying portal screen has different scope from the one of portlet, it cannot be gotten from the processing within the portlet.
By configuring this Action handler, it becomes possible to use for portlets the request parameter when displaying portal screen.

4.7 Event processing (processEvent cycle)

Since there is no screen display in processEvent cycle, only an Event handler class implemented with PortletActionHandler interface will be created.
Moreover, Event processing will be executed by registering the Event handler created in the portlet administration screen and configuring the event through Action processing of any portlet.

RenderParamter configuration, PortletPreferences configuration and update processing of such as event configuration are handled by Event processing.

4.7.1 Creating Event handler

An example of Event handler which configures RenderParameter is shown below.

1. Create a Java class implemented with PortletEventHandler interface, and define the following function. (This Java class is called Event handler.)

(ex.) sample.portal.SampleEventHandler.java

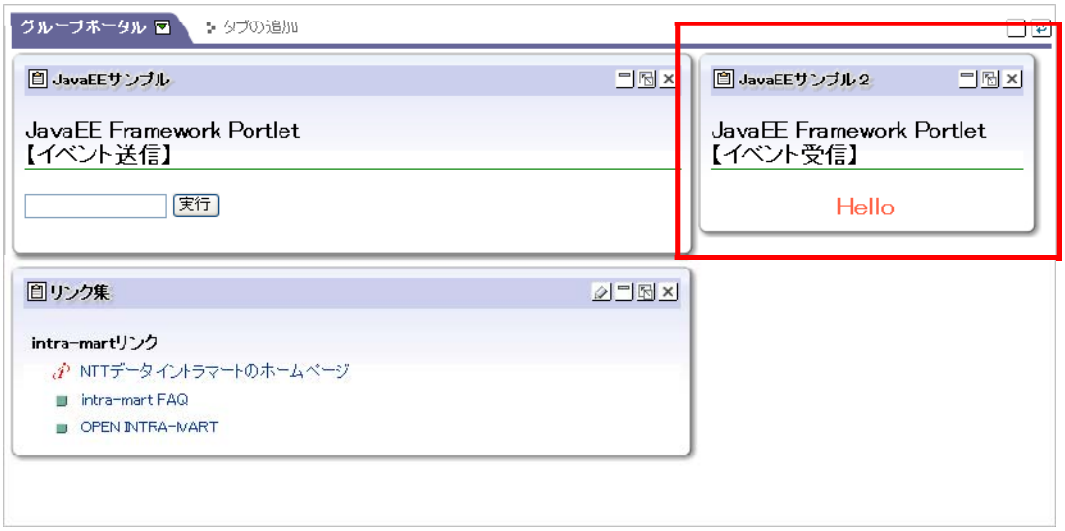
```
// Implement "PortletEventHandler" interface.
public class SampleEventHandler implements PortletEventHandler {

   public Serializable handleEvent(String portletCd, EventRequest request, EventResponse response) {

      // Get the event object.
      // The event object used among portlets for intramart is "ImEvent".
      ImEvent event = (ImEvent) request.getEvent().getValue();

      // Configure the request parameter used for screen display.
      response.setRenderParameter(event.getEventId(), event.getEvent());

      return null;
  }
}
```

- For Event handler, a function container that has been created with Script Development Model used for render may be used.
- By configuring the return value, more events can be made occur.
  * Pay attention to the design so as not to cause loops.

2. Register the Event handler to the portlet.

<ポートレット新規登録画面>

3. Locate the screen created as described in [4.5 Screen development (render cycle)] and the portlet which can call the event created as described in [4.6 Action processing (processAction cycle)エラー! ブックマーク名が指定されていません。エラー! ブックマーク名が指定されていません。] in the portal screen, and call the Action processing from the portlet for configuring the event.
Event handler will be executed from the Action processing, and then the screen will be displayed again.



< Portal screen – Event processing >

4.7.2 ImEvent object

Among intra-mart portlets, ImEvent object is configured as the Event object which can be gotten from EventRequest. Im Event object contains the following information:

Property name	Type	Value
id	String	Event ID configured by the Action processing. If omitted, character string “ImEvent” will be configured.
value	String	Event value configured by the Action processing. This will be gotten as character string even when configuring an object.
source	String	Portlet code of the portlet which configured the Event.

4.7.3 EventRequest, EventResponse

Arguments of Event handler are the dedicated objects for Event processing, i.e. EventRequest object and EventResponse object.

By using these objects, it is possible to get a request argument and to configure an event.

For details, refer to “Portlet API 2.0 Document”.

4.7.4 RenderParameter configuration

The request parameter which can be gotten when displaying the screen (RenderParameter) is configured in Action processing or Event processing.

The RenderParameter which had already been configured at the time of calling the Event processing has been contained in EventRequest, and will be automatically inherited to render cycle unless specifically handled. If inheritance is not required, it needs to be explicitly deleted.

- In the specification of JSR286, RenderParameter will not be automatically inherited. It should be noted that this is a specification unique to the intra-mart portlet containers.

(ex.) Configuring the RenderParameter in processEvent cycle

```
// When configuring the request parameter:
response.setRenderParameter("param1", "value1");

// When deleting the request parameter:
response.setRenderParameter("param2", null);
```

(ex.) Using the RenderParameter in render cycle

```
request.getParamter(eventId);
```

4.7.5 PortletPreferences configuration

PortletPreferences means portlet information storage area for each user.

In the portlet container of intra-mart, information configured in the PortletPreferences will be stored in the database.

Information configured here can be referred at any time while the screen is displayed.

(ex.) Using java.portlet.PortletPreferences

```
// Get the PortletPreferences.
PortletPreferences preferences = request.getPreferences();

// Get the information configured in the PortletPreferences.
String value = preferences.getValue("key1", "Default Value");

// Configure the information in the PortletPreferences.
preferences.setValue("key1", "value1");

// Determinate the PortletPreferences.
// Unless this processing is executed, information will not be stored.
preferences.store();
```

4.7.6 Event configuration

By configuring an event, it is possible to pass over its handling to multiple portlets other than the one which accepted the Event processing, through notifying the occurrence of the event.

However, doing event processing further in the event processing would result in lowering the performance, as well as making event loop occur. So pay sufficient attention when implementing.

(ex.) Configuring an event in processEvent cycle

```
response.setEvent(eventId, eventValue);
```

4.8 General-purpose New-Arrived-Portlet

In intra-mart portal module, portlet for displaying general-purpose newly arriving information has been made available. This general-purpose New-Arrived-Portlet displays the following items as newly arriving information:

- Category for indicating the type of newly arriving information
- Content of the newly arriving information (outline)
- Arrived date
- Information about the transition destination tied to the newly arriving information (link)

In order to display unique newly arriving information in the general-purpose New-Arrived-Portlet, the portlet should be registered according to the procedure as described below.

1. Implementing a provider for getting newly arriving information.

Information to be displayed in the New-Arrived-Portlet will be gotten from the provider implemented with the following interface:

- Interface for the provider of general-purpose newly arriving information:
  jp.co.intra_mart.foundation.portal.general_purpose_portlet.provider.NewArrivedProvider

For portal modules, “Important notice” portlet has been made available as the one which uses general-purpose New-Arrived-Portlet, and it displays information on password administration history by using the following provider. This would be a reference for implementing unique providers.

- Provider of general-purpose newly arriving information on password history administration:
  jp.co.intra_mart.foundation.security.password.PasswordHistoryNewArrivedProvider

For details on interface of the provider and on implementing the provider, refer to “Portal API List”.

2. Defining the implementation of the provider.

Information about the provider class implemented will be defined by creating the configuration file in an arbitrary location under Storage Service. Define the following information in the configuration file:

< Provider configuration file >

Element	Explanation
new-arrived-portlet/sort	Defines whether the newly arriving information which has been gotten should be sorted by category.
new-arrived-portlet/provider/provider-class	Defines the implementation class of the provider.
new-arrived-portlet/provider/init-param	Defines the initial parameter to be configured to the provider class.

Below is an example of describing the configuration file.

<sample_portlet.xml>

```
<new-arrived-portlet>
  <sort>true</sort>
  <provider>
    <provider-class>
      sample.SampleProvider
    </provider-class>
    <init-param>
      <param-name>param1</param-name>
      <param-value>hogehoge</param-value>
    </init-param>
  </provider>
</new-arrived-portlet>
```

By registering more than one provider, it is also possible to display multiple newly arriving information in one portlet.
For the detailed definition of configuration file, refer to “intra-mart WebPlatform/AppFramework Portal Configuration Guide”.

3. Registering the portlet.

Register the portlet for displaying newly arriving information from the group administrator’s [Portal] – [Portlet] editing screen.
When configuring general-purpose New-Arrived-Portlet, it is necessary to configure “Page type”, “Screen path or URL type” and “Page argument”. For example, configuration in the sample is as follows:

- Page type: Servlet/JSP
- Screen path or URL type: /portal/portlets/newly_arrived_view.jsp
- Page argument: page_param=/portal/sample_portlet.xml
- Action handler (option):
  jp.co.intra_mart.foundation.portal.general_purpose_portlet.handler.NewArrivedPortletHandler

The JSP to be specified as “Screen path or URL type” should be the one such as for displaying the newly arriving information which the provider has gotten. The JSP specified in the sample is a default implementation provided by the portal module.
For “Page argument”, path of the configuration file which defined the provider information should be specified in “page_param= ...” format. For the value of page_param, path under the root directory of Storage Service should be specified.

4.8.1 Using edit mode

Page type of the general-purpose New-Arrived-Portlet is “JSP/Servlet”, and edit mode is enabled by default.
However, in order to actually use edit mode, it is necessary to configure the Action handler as specified in the previous section, when registering the portlet.

When switching to edit mode, a screen as shown below will be displayed.
It is a feature only to switch between sort items and between ascending/descending order of the view mode. Usually a feature like this can be realized solely by the view mode, but this screen is incorporated with an implication as a sample of edit mode.

< Edit mode of general-purpose New-Arrived-Portlet >

This feature consists mainly of the following modules:

- Mode switching module: As there is just one page path which can be registered, processing to determine the mode and to switch is required.
  - "portal/portlets/ newly_arrived_view.jsp"
- Displaying module for view mode: This handles screen display in view mode.
  - "portal/portlets/ newly_arrived /newly_arrived_view.jsp"
- Displaying module for edit mode: This handles screen display in edit mode.
  - "portal/portlets/ newly_arrived /newly_arrived_edit.jsp"
- Action handler for configuration: This handles processing when the [Configuration] button is clicked. As well as data registration, this handles mode switching after registration.
  - "jp.co.intra_mart.foundation.portal.general_purpose_portlet.handler.NewArrivedPortletHandler"

These sources would be references for creating portlets.

4.8.2 "Important notice" portlet

"Important notice" portlet is a uniquely created portlet application supporting JSR286, and its screen display processing is implemented by using the mechanism of general-purpose New-Arrived-Portlet.
Currently the following provider classes created by JavaEE Development Model are used, and information on notice from the system administrator and on password history administration will be displayed.

- jp.co.intra_mart.foundation.security.password.PasswordHistoryNewArrivedProvider
- jp.co.intra_mart.foundation.portal.portlets.system_notice.SystemNoticeProvider

Information to be displayed in the important notice is defined by the following file.

- <root directory of Storage Service>/portal/system_notice.xml

By implementing the provider class with Script Development Model and adding definition to the configuration file, it becomes possible to deal with information other than as described above, as "important notice".

< Definition file: system_notice.xml >

```
<new-arrived-portlet>
  <sort>true</sort>
  <provider>
```

```
    <provider-class>
      jp.co.intra_mart.foundation.security.password.PasswordHistoryNewArrivedProvider
    </provider-class>
  </provider>
  <provider>
    <provider-class>
      jp.co.intra_mart.foundation.portal.portlets.system_notice.SystemNoticeProvider
    </provider-class>
  </provider>
</new-arrived-portlet>
```

4.8.3 Implementing a provider by using Script Development Model

For portal modules, the following implementation class of provider for getting newly arriving information from Script Development Model has already been made available.

- jp.co.intra_mart.foundation.portal.general_purpose_portlet.provider.PageBaseCallProvider

Providers of JavaEE Development Model and of Script Development Model can be present together, so by implementing the provider class with Script Development Model and adding definition to the configuration file, it becomes possible to add general-purpose newly arriving information to be displayed in the portlet.

< Example of defining a provider by using Script Development Model >

```
<new-arrived-portlet>
  <sort>true</sort>
  <provider>
    <!-- Provider class for Script Development Model. This should be fixed. -->
    <provider-class>
      jp.co.intra_mart.foundation.portal.general_purpose_portlet.provider.PageBaseCallProvider
    </provider-class>
    <init-param>
      <!-- Define the provider created with Script Development Model. -->
      <param-name>pagePath</param-name>
      <param-value>/hoge/sample_provider.js</param-value>
    </init-param>
  </provider>
</new-arrived-portlet>
```

5 Portlet development (JSP/Servlet)

5.1 Outline

JSP/Servlet portlet is created by using JSP/Servlet for screen display processing. It can be created in the same way as developing regular screen display.

In order to realize portlet lifecycle, Action handler and Event handler can be created, and the methods to create them are not different from those of creating with JavaEE Framework.

Refer to [4.6 Action processing (processAction cycle)] and [4.7 Event processing (processEvent cycle)] in the chapter on JavaEE development.

6 Portlet development (Java portlet)

6.1 Screen development

Java portlet has been defined as a module similar to Servlet, and is created in the same way as Web application which uses Servlet, and is used upon being deployed in a portal server as WAR file.
However, information on the portlet definition is described in portlet.xml under WEB-INF.
Apart from that it will be developed in the same way as creating regular Web application, but because of not being Servlet, frameworks like Jakarta Struts which use Servlet cannot be used.

How to create a Java portlet is briefly described below.
For details on Java portlet (JSR168, JSR286), refer to specialized documents and the API document (Portlet API 2.0).

- http://jcp.org/en/jsr/detail?id=168 (Page on JSR168 specifications)
- http://jcp.org/en/jsr/detail?id=286 (Page on JSR286 specifications)

1. Creating a portlet class

Java portlet needs to be implemented with javax.portlet.Portlet interface.
However, for PortletAPI, GenericPortlet class implemented with basic processing for creating portlets easily has already been made available.

```
package sample.portlet;
import javax.portlet.GeneralPortlet;

// Inherit GenericPortlet.
public class SamplePortlet extends GenericPortlet {

  // In GenericPortlet, display processing methods are called for each mode.
  // Describe processing in VIEW mode.
  protected void doView(RenderRequest request, RenderResponse response)

      // Describe processing in VIEW mode.

    // Call JSP.
    // Instead of ServletContext, use PortletContext.
    PortletContext context = getPortletContext();
    // Instead of RequestDispatcher, use PortletRequestDispatcher.
    PortletRequestDispatcher rd = context.getRequestDispatcher("/WEB-INF/jsp/sample_viewjsp");
    // In portlet, forward method cannot be used. Only Include can be used.
    rd.include(request, response);
  }

  //Describe processing in EDIT mode.
  protected void doEdit(RenderRequest request, RenderResponse response)

      // Describe processing in EDIT mode.
  }

  // Describe processing in HELP mode.
  protected void doHelp(RenderRequest request, RenderResponse response)

      // Describe processing in HELP mode.
  }
```

```
   // Describe processing in submit.
   public void processAction(ActionRequest request, ActionResponse response)

         // Describe processing in submission.
         // After this, display method will be automatically called and therefore calling JSP is not required.
   }

}
```

2. Creating JSP

```
// Portlet tag library can be used.
<%@ taglib uri="http://java.sun.com/portlet" prefix="portlet" %>

<%
   // In portlet, instead of HttpServletRequest, HttpServletResponse,
   // use RenderRequest, RenderResponse.
   // How to use them are almost same.
   RenderRequest rRequest = (RenderRequest)request.getAttribute("javax.portlet.request");
   RenderRespnse rResponse = (RenderResponse)request.getAttribute("javax.portlet.response");

   // Get the parameter from the request.
   String param = rRequest.getParameter("param");


   // Get the URL for redisplaying the portal screen.
   // Getting the URL for redisplaying the portal screen (RenderURL).
   PortletURL renderURL = rResponse.createRenderURL();

   // Getting the URL for redisplaying the portal screen after handling submit processing (Action URL).
   PortletURL actionURL = rResponse.createActionURL();
%>

<!-- Call a service to handle some registration processing -->
<!—Configure the Action URL -->
<!-- Specify the window with target attribute -->
<form action="<%= actionURL.toString() %>" name="sample_portlet_form" target="IM_MAIN" method="POST">

   <!-- Values used in the registration processing -->
   <input type="text" name="param" value="">

   <input type="submit" value="execute">
</form>
```

- Since ActionURL contains portal information, its data amount will increase accordingly. POST should be specified as the method.

3. Creating portlet.xml

In portlet.xml, describe information on portlet definition.

For details on how to describe, refer to the documents on JSR168 or JSR286 available from the following URLs:

- http://jcp.org/en/jsr/detail?id=168
- http://jcp.org/en/jsr/detail?id=286

```
<?xml version="1.0" encoding="UTF-8"?>
<portlet-app
    xmlns="http://java.sun.com/xml/ns/portlet/portlet-app_2_0.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/portlet/portlet-app_2_0.xsd
    http://java.sun.com/xml/ns/portlet/portlet-app_2_0.xsd"
    version="2.0">

    <portlet>
        <portlet-name>SamplePortlet</portlet-name>
        <portlet-class>sample.SamplePortlet</portlet-class>
        <expiration-cache>300</expiration-cache>
        <supports>
            <mime-type>text/html</mime-type>
        </supports>
        <supported-locale>ja</supported-locale>
        <portlet-info>
            <title>Sample Portlet</title>
        </portlet-info>
    </portlet>

</portlet-app>
```

< portlet.xml for JSR286 >

4. Creating web.xml

Although it is not necessary to describe in web.xml, an empty file should be contained to be deployed on AP server as a Web application.

It is also possible to describe additional information, but in such a case description in web.xml may differ depending on the Servlet API.

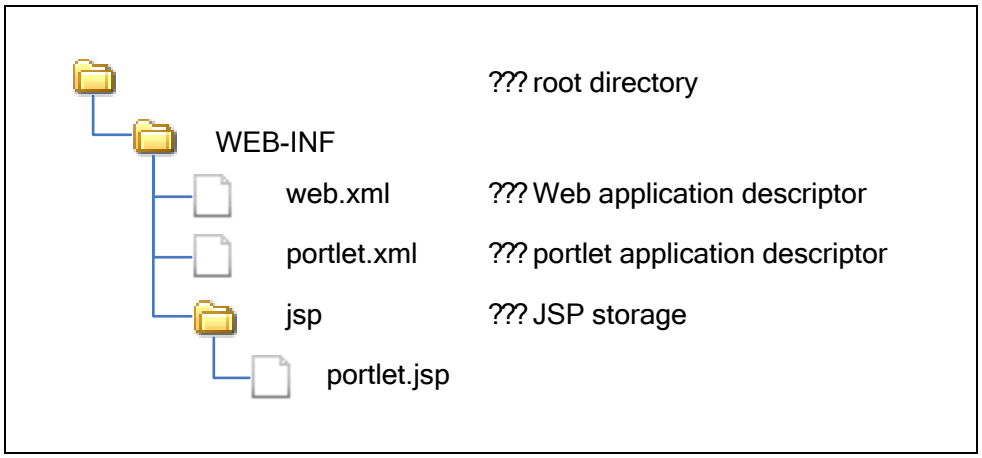
Version definition information should not be omitted, so that version information can be gotten.

- Servlet2.3: Specified by DOCTYPE
- Servlet2.4 and newer: Specified by XMLSchema

```
<?xml version="1.0" encoding="UTF-8"?>

<!—In case of Servlet 2.4 -->
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
        version="2.4">

</web-app>
```

5. Creating WAR file

By bringing created files together and creating WAR file, a portlet application will be completed.
In WEB-INF, two deployment descriptors of web.xml and portlet.xml are stored.
Also, in Java portlet it is defined that a portlet shall not be called directly, whereas if it is located on AP server it may be possible to access directly to JSP.
Therefore, in a portlet application JSP is usually stored under WEB-INF.



< Structure of WAR file >

6.2 Action processing (processAction cycle)

In the previous section, how to create a portlet is described narrowing down to screen display processing.
This section explains about several things which can be done in processAction.
A sample of processAction is shown below.

(ex.) Sample of processAction

```
// Implement “PortletActionHandler” interface.
public class SamplePortlet extends GenericPortlet {

  …

   public void processAction(ActionRequest request, ActionResponse response) {

   // Get the request argument.
   String value = request.getParameter("param1");

   // Configure the event.
   // Use as the key the QName specified in portlet.xml.
   response.setEvent(new QName("sampleNamespace", "SampleEvent"), value);

  }
}
```

6.2.1 ActionRequest, ActionResponse

Arguments of processAction are the dedicated objects for this method, i.e. ActionRequest object and ActionResponse object.

By using these objects, it is possible to get a request argument and to configure an event.

For details, refer to “Portlet API 2.0 Document”.

6.2.2 Event configuration

In the sample shown above, an event is configured to ActionResponse, but it is necessary to describe in portlet.xml whether the event can be configured or not, and what kind of events can be configured.

(ex.) Sample of definition of a transmittable event

```
<portlet-app
      xmlns="http://java.sun.com/xml/ns/portlet/portlet-app_2_0.xsd"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://java.sun.com/xml/ns/portlet/portlet-app_2_0.xsd
      http://java.sun.com/xml/ns/portlet/portlet-app_2_0.xsd"
      version="2.0">

      <portlet>
            <portlet-name>SamplePortlet</portlet-name>
            <portlet-class>sample.SamplePortlet</portlet-class>
            <expiration-cache>300</expiration-cache>
            <supports>
                  <mime-type>text/html</mime-type>
            </supports>
            <supported-locale>ja</supported-locale>
            <portlet-info>
                  <title>Sample Portlet</title>
            </portlet-info>
            <supported-publishing-event>
                  <!-- Definition of the event, among those defined by event-definition, which can be transmitted
by this portlet -->
                  <qname xmlns:im_portal="urn: sampel:event">
                     sample:SampleEvent
                  </qname>
            </supported-publishing-event>
      </portlet>

    <event-definition>
        <!-- Definition of the event which can be used. -->
        <qname xmlns:sample='urn:sampel:event'>sample:SampleEvent</qname>
        <!-- Specifying the type of event object -->
        <value-type>java.lang.String</value-type>
    </event-definition>

</portlet-app>
```

6.3 Event processing (processEvent cycle)

This section explains about processing of processEvent cycle.

A sample of processEvent is shown below.

(ex.) Sample of processEvent

```
// Implement “PortletEventHandler” interface.
public class SamplePortlet extends GenericPortlet {

 ~

   public void processEvent(EventRequest request, EventResponse response) {

     // Get the event object.
     // Type of the event object is the one specified in portlet.xml.
     String event = (String) request.getEvent()

     // Configure the request parameter used for screen display.
     response.setRenderParameter("param1", event);
  }
}
```

6.3.1 EventRequest, EventResponse

Arguments of processEvent are the dedicated objects for this method, i.e. EventRequest object and EventResponse object.

By using these objects, it is possible to get a request argument and to configure an event.

For details, refer to “Portlet API 2.0 Document”.

6.3.2 Event configuration

In the sample shown above, the event is gotten from EventResponse, but it is necessary to describe in portlet.xml what kind of events can be received.

(ex.) Sample of definition of a receivable event

```
<portlet-app
    xmlns="http://java.sun.com/xml/ns/portlet/portlet-app_2_0.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/portlet/portlet-app_2_0.xsd
    http://java.sun.com/xml/ns/portlet/portlet-app_2_0.xsd"
    version="2.0">

    <portlet>
        <portlet-name>SamplePortlet</portlet-name>
        <portlet-class>sample.SamplePortlet</portlet-class>
        <expiration-cache>300</expiration-cache>
        <supports>
            <mime-type>text/html</mime-type>
        </supports>
        <supported-locale>ja</supported-locale>
        <portlet-info>
            <title>Sample Portlet</title>
        </portlet-info>
        <supported-processing-event>
            <!-- Definition of the event, among those defined by event-definition, which can be received by
this portlet -->
            <qname xmlns:im_portal="urn:sample:event">
                sample:SampleEvent
            </qname>
        </supported-processing-event>
    </portlet>

    <event-definition>
        <!-- Definition of the event which can be used. -->
        <qname xmlns:sample='urn:sampel:event'>sample:SampleEvent</qname>
        <!-- Specifying the type of event object -->
        <value-type>java.lang.String</value-type>
    </event-definition>

</portlet-app>
```

7 Notes

7.1 Notes on creating a screen

Portlet is also a kind of Web application, so creating it is basically not different from the ordinary developing methods.
However, since consequentially portlet screen is displayed as a part of portal screen, there are several restrictions on crating it as follows:

1. Portlet is displayed as a part of HTML, and therefore the following tags which represent HTML structure cannot be used. Only contents of the document indicated within the area of <BODY> tag should be described.
   Some browsers may display portlets even if describing the following tags, but such description is not recommended.
   - <HEAD>
   - <BODY>
   - <!DOCTYPE>

2. Screen style is specified in the portal screen. As noted above, since <HEAD> tag cannot be used, definition of style class cannot be described. Although it is possible to configure the style individually for each tag, screens should be created as simply as possible, not depending on styles.

   - Specifying the style by using <STYLE> tag and <LINK> tag may have effect on the layout of portal screen.
   - If <IMART type="imDesignCss"> / < imarttag:imartDesignCss> tags are used, color pattern will be changed dynamically, whereas if cache mechanism of the portlet is set as valid, change of the color pattern will not be reflected.
     These tags are specified in the portal screen and can be used without being specified repeatedly, so these should not be described for portlets.

3. It is necessary to define the names of object, variable and function to be used for JavaScript with unique names. This also applies to ordinary screen development; for portlets particular attention need to be given as it is necessary to arrange so that they should not overlap in every portlet on the portal screen. In addition, when using published common JavaScript, attention need to be given as version discrepancy may cause unforeseen operation. In portal screen the following common libraries are used, so libraries of the same version should be used for portlets.
   - prototype.js v1.6.0.3
   - script.aculo.us（builder.js, effect.js, dragdrop.js, slider.js) v1.8.2

4. It is necessary to define ID attribute of tags and NAME attributes of controls with unique names. It is recommended that for example, every ID would be provided with a prefix assigned to the corresponding application.
   In addition, the following prefixes cannot be used because they are used by the system and by applications provided from intra-mart.
   - "IM_"
   - "_" (underscore)

5. Because several pages are displayed in portal screen, if a page which requires more processing time it could take accordingly more time to display the portal screen.
   It is recommended that simple screens should be created as far as possible.

6. In portal screen, when a portlet is maximized a processing which maximizes the height of content in accordance with the portlet is executed.
   But if the content consists of multiple elements, which element should be maximized cannot be controlled and therefore unintended result may occur.
   In such a case, the screen should be created so that its element is just one, by means of, for example, enclosing every content with element.

8 Sample

8.1 About samples

The intra-mart WebPlatform/AppFramework have samples included to explain how to use Action processing and Event processing, the features specific to portlets.

By using these samples, processing for Action handler and for Event handler can be confirmed.
Specify the transmission destination by “Event transmission” portlet, and then confirm that a message is generated.
In addition, it is recommended to confirm configuration of Action handler/Event handler in the portlet editing screen.

8.2 Configuring samples

If installation is carried out with samples being set valid, sample programs and sample data will be installed.
In order to use a portal sample, it is necessary to import portal data in addition.
Log in as a login group administrator, and from the administration menu select [Portal] – [Import/Export], and then import the following sample data:

- sample/portal/portal_event_sample.zip (if character encoding of server is Windows-31J)
- sample/portal/portal_event_sample_u8.zip (if character encoding of server is UTF-8)

By importing sample data, “Sample” tab will be created in the group portal.
In “Sample” tab the following portlets are placed:

- **im-JavaEE Service Framework portlet**
  - event transmission (JavaEE): executes an Action and make an event occur.
  - event reception 1 (JavaEE): receives the relevant event and displays the message.
  - event reception 2 (JavaEE): receives the relevant event and displays the message.

- **Presentation Page portlet**
  - event transmission (JSSP): executes an Action and make an event occur.
  - event reception 1 (JSSP): receives the relevant event and displays the message.
  - event reception 2 (JSSP): receives the relevant event and displays the message.

This sample data depends on login group, and therefore unless login group ID is “default”, “Sample” tab will not be displayed. If “Sample” tab is not displayed, create a sample portal with the following procedure:

1. In the portal screen, click the [Add tab] button and create new tab named “Sample”.
2. From the context menu, select [Add portlet].
3. Select [Application] – [Event sample], and add all the portlets searched to the portal screen.

8.3 List of sample files

File path	Explanation
<% IM_ROOT %>/doc/imart/sample/portal/event	Sample JSP file for JavaEE Development Model
<% IM_ROOT %>/doc/imart/WEB-INF/classes/sample/portal/event/handler	Sample handler class for JavaEE Development Model
<% IM_ROOT %>/pages/src/sample/portal/event	Sample JSSP file for Script Development Model
<% IM_ROOT %>/storage/sample/portal	Import data for “Sample” portal screen

intra-mart WebPlatform/AppFramework Ver.7.2
Portlet Programming Guide

First Edition: 2012/05/07

TEL: 03-5549-2821
FAX: 03-5549-2816
E-MAIL: info@intra-mart.jp
URL: http://www.intra-mart.jp/