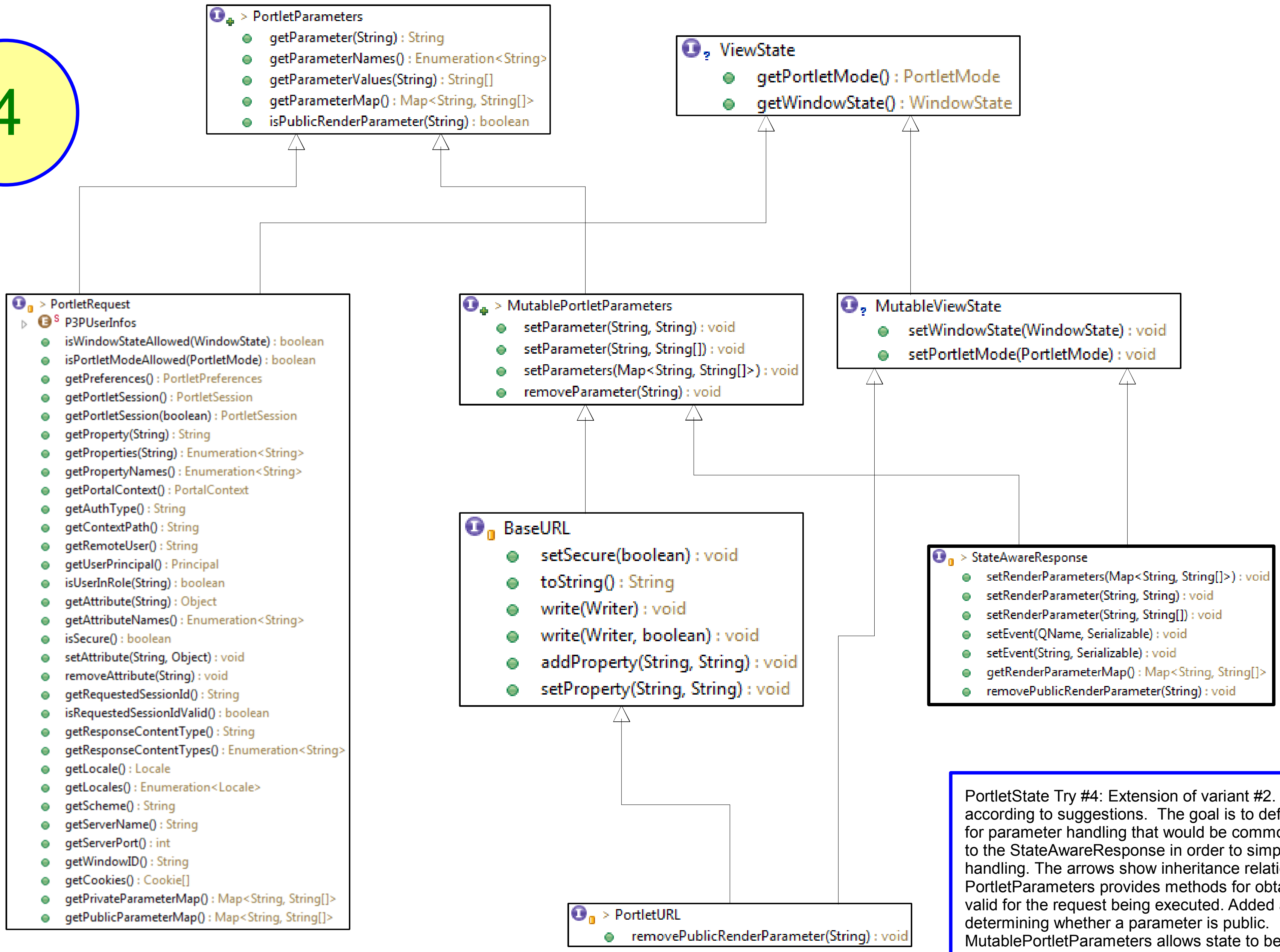
Goals / Use Cases

- Provide common interface for parameter handling on URLs and responses
  - Makes it easier to write state handling code
- Provide clear separation between the render parameters, which represent portlet state, and the action or resource parameters, which provide input for execution of a single request
- Improve usability of the portlet parameter handling interfaces
  - Increase regularity of the parameter handling method definitions and behavior across the four requests – render, action, resource, and event
  - Reduce the number of method calls necessary to handle state in a portlet
  - Resulting method definitions should behave intuitively. The developer should be able to understand the interfaces from the apidocs alone. A separate document describing the exceptions and special handling should not be necessary.
  - Eliminate the need to define the same method in multiple interfaces to avoid duplicating descriptions

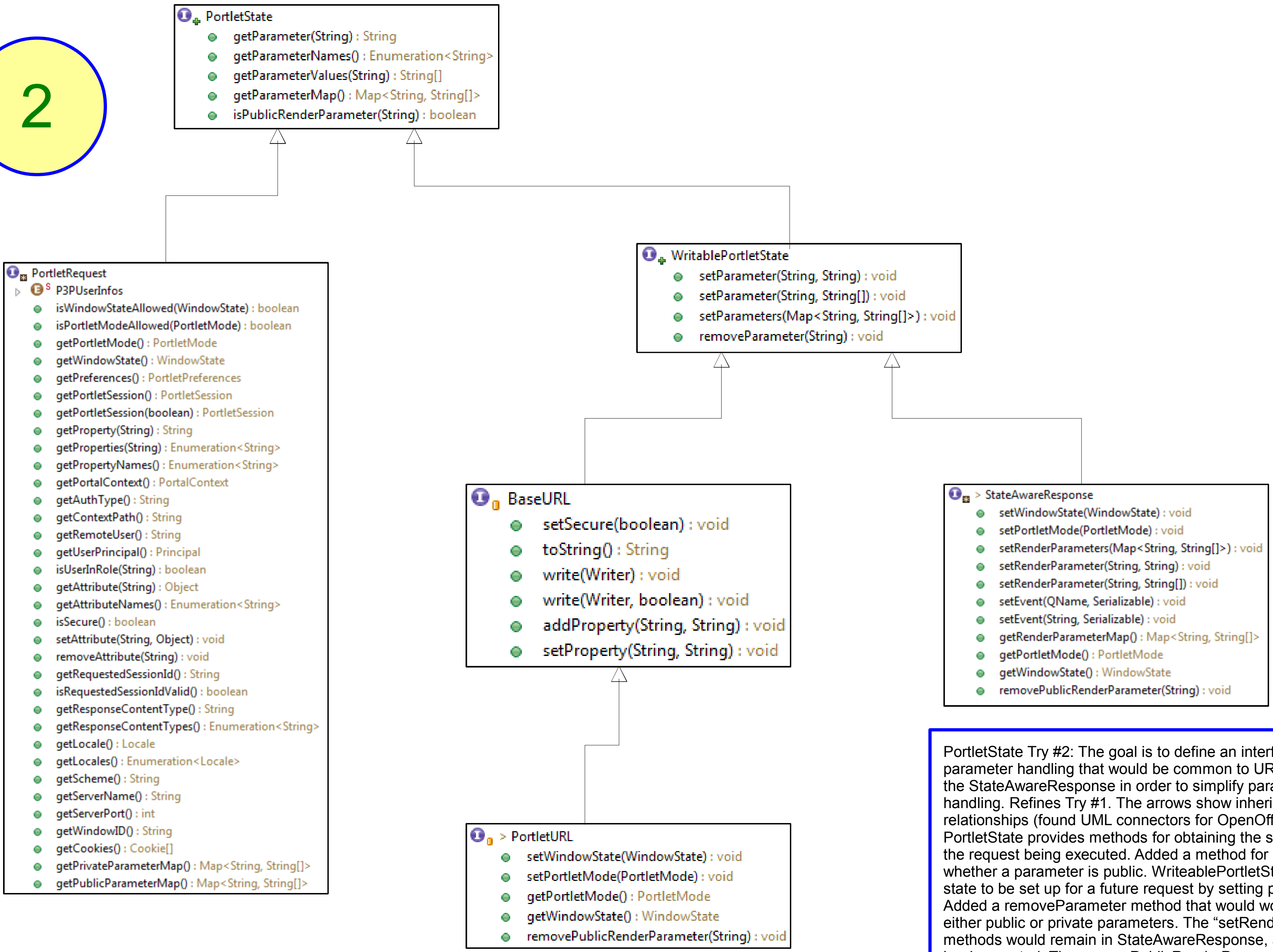
4


PortletParameters
getParameter(String) : String
getParameterNames() : Enumeration<String>
getParameterValues(String) : String[]
getParameterMap() : Map<String, String[]>
isPublicRenderParameter(String) : boolean
ViewState
getPortletMode() : PortletMode
getWindowState() : WindowState
PortletRequest
P3PUserInfos
isWindowStateAllowed(WindowState) : boolean
isPortletModeAllowed(PortletMode) : boolean
getPreferences() : PortletPreferences
getPortletSession() : PortletSession
getPortletSession(boolean) : PortletSession
getProperty(String) : String
getProperties(String) : Enumeration<String>
getPropertyNames() : Enumeration<String>
getPortalContext() : PortalContext
getAuthType() : String
getContextPath() : String
getRemoteUser() : String
getUserPrincipal() : Principal
isUserInRole(String) : boolean
getAttribute(String) : Object
getAttributeNames() : Enumeration<String>
isSecure() : boolean
setAttribute(String, Object) : void
removeAttribute(String) : void
getRequestedSessionId() : String
isRequestedSessionIdValid() : boolean
getResponseContentType() : String
getResponseContentTypes() : Enumeration<String>
getLocale() : Locale
getLocales() : Enumeration<Locale>
getScheme() : String
getServerName() : String
getServerPort() : int
getWindowID() : String
getCookies() : Cookie[]
getPrivateParameterMap() : Map<String, String[]>
getPublicParameterMap() : Map<String, String[]>
MutablePortletParameters
setParameter(String, String) : void
setParameter(String, String[]) : void
setParameters(Map<String, String[]>) : void
removeParameter(String) : void
MutableViewState
setWindowState(WindowState) : void
setPortletMode(PortletMode) : void
BaseURL
setSecure(boolean) : void
toString() : String
write(Writer) : void
write(Writer, boolean) : void
addProperty(String, String) : void
setProperty(String, String) : void
StateAwareResponse
setRenderParameters(Map<String, String[]>) : void
setRenderParameter(String, String) : void
setRenderParameter(String, String[]) : void
setEvent(QName, Serializable) : void
setEvent(String, Serializable) : void
getRenderParameterMap() : Map<String, String[]>
removePublicRenderParameter(String) : void
PortletURL
removePublicRenderParameter(String) : void

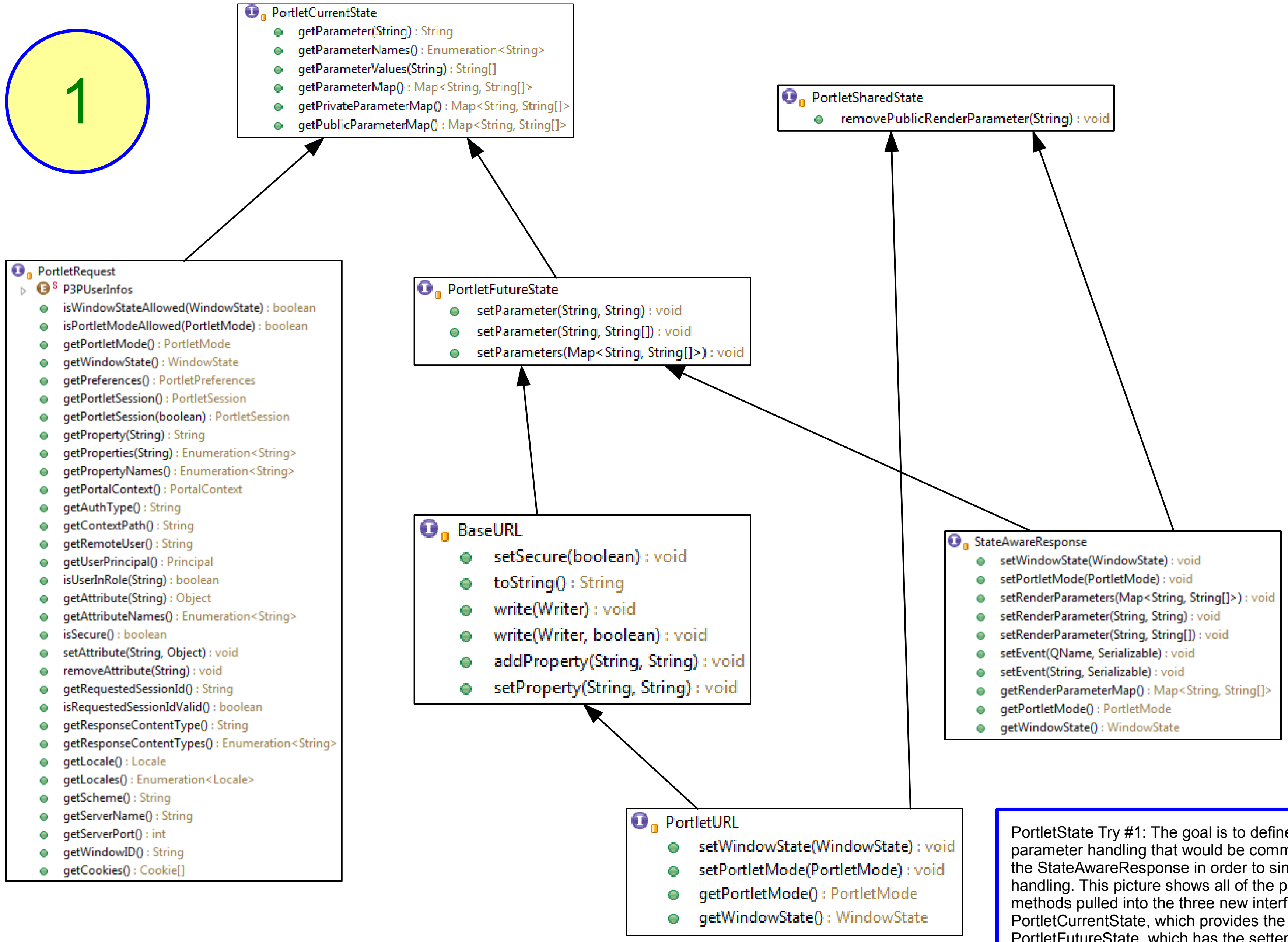
PortletState Try #4: Extension of variant #2. Refactored according to suggestions. The goal is to define an interface for parameter handling that would be common to URLs and to the StateAwareResponse in order to simplify parameter handling. The arrows show inheritance relationships. PortletParameters provides methods for obtaining the state valid for the request being executed. Added a method for determining whether a parameter is public. MutablePortletParameters allows state to be set up for a future request by setting parameters. Added a removeParameter method that would work with either public or private parameters. The “setRender...” methods would remain in StateAwareResponse, and would be deprecated. The removePublicRenderParameter method would remain in PortletURL and StateAwareResponse and would be deprecated.
Abstracted portlet mode and window state into ViewState and MutableViewState interfaces.

Earlier Variants

2


PortletState
getParameter(String) : String
getParameterNames() : Enumeration<String>
getParameterValues(String) : String[]
getParameterMap() : Map<String, String[]>
isPublicRenderParameter(String) : boolean
PortletRequest
P3PUserInfos
isWindowStateAllowed(WindowState) : boolean
isPortletModeAllowed(PortletMode) : boolean
getPortletMode() : PortletMode
getWindowState() : WindowState
getPreferences() : PortletPreferences
getPortletSession() : PortletSession
getPortletSession(boolean) : PortletSession
getProperty(String) : String
getProperties(String) : Enumeration<String>
getPropertyNames() : Enumeration<String>
getPortalContext() : PortalContext
getAuthType() : String
getContextPath() : String
getRemoteUser() : String
getUserPrincipal() : Principal
isUserInRole(String) : boolean
getAttribute(String) : Object
getAttributeNames() : Enumeration<String>
isSecure() : boolean
setAttribute(String, Object) : void
removeAttribute(String) : void
getRequestedSessionId() : String
isRequestedSessionIdValid() : boolean
getResponseContentType() : String
getResponseContentTypes() : Enumeration<String>
getLocale() : Locale
getLocales() : Enumeration<Locale>
getScheme() : String
getServerName() : String
getServerPort() : int
getWindowID() : String
getCookies() : Cookie[]
getPrivateParameterMap() : Map<String, String[]>
getPublicParameterMap() : Map<String, String[]>
WritablePortletState
setParameter(String, String) : void
setParameter(String, String[]) : void
setParameters(Map<String, String[]>) : void
removeParameter(String) : void
BaseURL
setSecure(boolean) : void
toString() : String
write(Writer) : void
write(Writer, boolean) : void
addProperty(String, String) : void
setProperty(String, String) : void
StateAwareResponse
setWindowState(WindowState) : void
setPortletMode(PortletMode) : void
setRenderParameters(Map<String, String[]>) : void
setRenderParameter(String, String) : void
setRenderParameter(String, String[]) : void
setEvent(QName, Serializable) : void
setEvent(String, Serializable) : void
getRenderParameterMap() : Map<String, String[]>
getPortletMode() : PortletMode
getWindowState() : WindowState
removePublicRenderParameter(String) : void
PortletURL
setWindowState(WindowState) : void
setPortletMode(PortletMode) : void
getPortletMode() : PortletMode
getWindowState() : WindowState
removePublicRenderParameter(String) : void

PortletState Try #2: The goal is to define an interface for parameter handling that would be common to URLs and to the StateAwareResponse in order to simplify parameter handling. Refines Try #1. The arrows show inheritance relationships (found UML connectors for OpenOffice, yay!) PortletState provides methods for obtaining the state valid for the request being executed. Added a method for determining whether a parameter is public. WriteablePortletState allows state to be set up for a future request by setting parameters. Added a removeParameter method that would work with either public or private parameters. The “setRender...” methods would remain in StateAwareResponse, and would be deprecated. The removePublicRenderParameter method would remain in PortletURL and StateAwareResponse and would be deprecated.
Question: Should PortletMode & WindowState be handled as “portlet state” in these interfaces? Or should we define an “ExtendedPortletState” containing the parameters, window state, and portlet mode?


1
PortletCurrentState
getParameter(String) : String
getParameterNames() : Enumeration<String>
getParameterValues(String) : String[]
getParameterMap() : Map<String, String[]>
getPrivateParameterMap() : Map<String, String[]>
getPublicParameterMap() : Map<String, String[]>
PortletSharedState
removePublicRenderParameter(String) : void
PortletRequest
P3PUserInfos
isWindowStateAllowed(WindowState) : boolean
isPortletModeAllowed(PortletMode) : boolean
getPortletMode() : PortletMode
getWindowState() : WindowState
getPreferences() : PortletPreferences
getPortletSession() : PortletSession
getPortletSession(boolean) : PortletSession
getProperty(String) : String
getProperties(String) : Enumeration<String>
getPropertyNames() : Enumeration<String>
getPortalContext() : PortalContext
getAuthType() : String
getContextPath() : String
getRemoteUser() : String
getUserPrincipal() : Principal
isUserInRole(String) : boolean
getAttribute(String) : Object
getAttributeNames() : Enumeration<String>
isSecure() : boolean
setAttribute(String, Object) : void
removeAttribute(String) : void
getRequestedSessionId() : String
isRequestedSessionIdValid() : boolean
getResponseContentType() : String
getResponseContentTypes() : Enumeration<String>
getLocale() : Locale
getLocales() : Enumeration<Locale>
getScheme() : String
getServerName() : String
getServerPort() : int
getWindowID() : String
getCookies() : Cookie[]
PortletFutureState
setParameter(String, String) : void
setParameter(String, String[]) : void
setParameters(Map<String, String[]>) : void
BaseURL
setSecure(boolean) : void
toString() : String
write(Writer) : void
write(Writer, boolean) : void
addProperty(String, String) : void
setProperty(String, String) : void
StateAwareResponse
setWindowState(WindowState) : void
setPortletMode(PortletMode) : void
setRenderParameters(Map<String, String[]>) : void
setRenderParameter(String, String) : void
setRenderParameter(String, String[]) : void
setEvent(QName, Serializable) : void
setEvent(String, Serializable) : void
getRenderParameterMap() : Map<String, String[]>
getPortletMode() : PortletMode
getWindowState() : WindowState
PortletURL
setWindowState(WindowState) : void
setPortletMode(PortletMode) : void
getPortletMode() : PortletMode
getWindowState() : WindowState

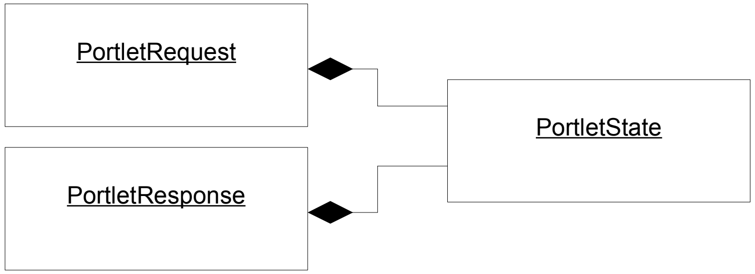
PortletState Try #1: The goal is to define an interface for parameter handling that would be common to URLs and to the StateAwareResponse in order to simplify parameter handling. This picture shows all of the parameter handling methods pulled into the three new interfaces PortletCurrentState, which provides the getter methods, PortletFutureState, which has the setter methods, and PortletSharedState, which contains the single method that works only with public render parameters. The arrows show an inheritance relationship. PortletCurrentState allows you to get the parameters associated with the request currently being executed, while PortletFutureState allows you to set up state for a future request. The future request would be a render request if the state is set on the StateAwareResponse. If set on a URL, it would be the state that would come into play if the URL is activated.

3-1

PortletState

getParameter(String)
getParameterNames()
getParameterValues(String)
getParameterMap()
isPublicRenderParameter(String)
setParameter(String, String)
setParameter(String, String[])
setParameters(Map<String, String[]>)
removeParameter(String)

PortletState Try #3: Implement concept of “Portlet State”. Define a regular interface for state handling for the portlet itself and for URLs. The state is represented by the public and private parameters that determine processing during a portlet request. It corresponds to the JSR 286 idea of portlet parameters.



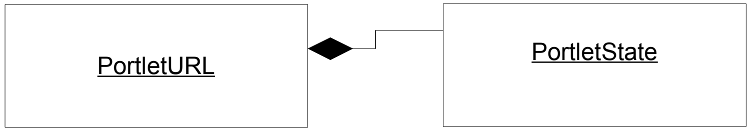
PortletRequest

New:
getPortletState()

Deprecated:
JSR 286 parameter methods

Each portlet request and each portlet response object has one portlet state object that they conceptually share. You use the getPortletState() method on the PortletRequest to obtain access to the PortletState object. Changes to the PortletState object obtained from the request are automatically reflected in the PortletResponse PortletState object and become the new portlet render state, so there is no need to explicitly set a new portlet state. The new state will be available during subsequent render requests.

Changing the portlet render state during processing of a render request or a resource request is not allowed, so an attempt to do so by using one of the setParameter... methods or by using the removeParameter() method will result in a javax.portlet.ReadOnlyException.

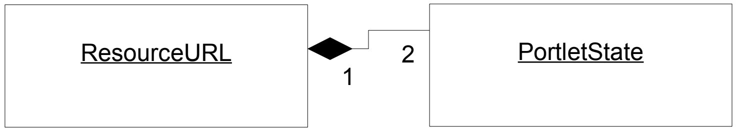


PortletURL

New:
getPortletState()

Deprecated:
JSR 286 parameter methods

Each portlet URL has one portlet state object. Depending on how the URL is created (new “create” method tbd), the PortletState for the URL will initially be either empty, or will be populated with the PortletState valid for the current request. You use the getPortletState() method to obtain access to the PortletState object. Changes to the PortletState object are automatically stored with the URL, so there is no need to explicitly set a new portlet state. The new state will be available during the portlet request initiated through this URL..



ResourceURL

New:
getRenderState()
getResourceState()

Deprecated:
JSR 286 parameter methods

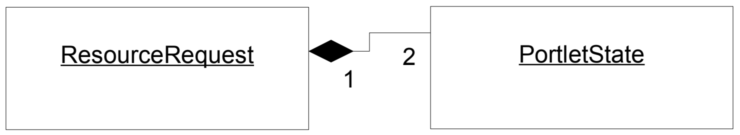
Each Resource URL has 1 or 2 portlet state objects.

The PortletState object representing the render state for the URL be populated with the render state valid for the current request. You use the getRenderState() method to obtain access to the read-only render state.

Changing the portlet render state on a ResourceURL is not allowed, so an attempt to do so will result in an InvalidStateException.

However, you can add additional state by using the getResourceState() method to obtain an empty PortletState object that allows additional resource parameters to be added.

3-2



ResourceRequest

New:
getPortletState()
getRenderState()
getResourceState()

Deprecated:
JSR 286 parameter methods

MimeResponse

New:
createActionURL(int copyFlag)
createRenderURL(int copyFlag)
createResourceURL(int copyFlag)

Deprecated:
None.

Each Resource Request has 1 or 2 portlet state objects.

The render state is the state that governed the render or resource request during which the resource URL leading to this resource request was generated. It is automatically copied to the resource URL when the URL is created and cannot be modified.

The resource state consists of additional state information that was appended to the URL after its creation. It is read-only and cannot be modified.

The PortletState obtained from the getPortletState() method is a read-only object representing the render state and resource state combined together. If a render state parameter and a resource state parameter have the same name, the resource parameter value(s) will appear in the value array before the render state value(s).

Note: I believe that it would be possible to add this new model while leaving the current JSR 286 model completely intact. We could specify that calling getPortletState() on a request activates the new model and deactivates the old model. That would avoid the problem of having to merge parameters set through the JSR 286 methods into the new PortletState model.

Although not strictly related to parameters, for completeness I wanted to show the proposal for new URL creation methods.

Three new create methods would be added, one for each URL type. The new methods would each contain a flag that would designate whether the the currently active portlet state should be copied into the new URL. The flag would have the following values:

COPY_NONE – no parameters would be copied; same behavior as the current createActionURL() and createRenderURL() methods.

COPY_RENDER_STATE – the currently active render state is copied to the URL.

COPY_PORTLET_STATE – the complete portlet state is copied to the new URL. During render request processing, this flag would have the same effect as the COPY_RENDER_STATE flag. However, during resource request processing, the portlet state represents the combined resource state and render state, so the result would be different