WindowId Proposal

Intro

Leonardo proposed the APIs after a discussion with Gerhard and Mark and based on the experience with MyFaces CODI (which was based on the experience with MyFaces Orchestra).
Originally we started at http://wiki.apache.org/myfaces/Drafts/WindowId

The purpose of the WindowId API is to explicitly establish an association between a client window and a UIComponent hierarchy rooted at a single UIViewRoot.

Terms

- **client window**. A client window may be a browser tab, a browser window, a pop-up or a portlet.
- **windowId**. A property of the client window that uniquely identifies a client window within the scope of a client session.

Invariants

- The lifetime of a windowId is smaller than a session but greater than an individual request processing lifecycle
- A window-id always is unique in the session of the user. A session contains multiple windowIds, but only exactly 1 window-id >per< browser tab.
- A client window is always associated with exactly one UIViewRoot instance at a time, but it may display many different UIViewRoots during its lifetime.
- The windowId must not be shared across browser tabs.
- The windowId must not be included in the ViewState, because it needs to be available before the ViewState has been decoded.
- The windowId must be the anchor to which the Flash is hitched.

Invariants (maybe postponed to JSF 2.3)

- The first request which results in a pages displayed to the user must work like all subsequent requests
  (e.g. the URL-mode adds the windowId to the URLs -> in this mode it isn't allowed to show the first page without the windowId in URL of the address-bar, because a refresh would lead to a different URL and the state which was submitted e.g. via Ajax requests would be lost.)
- Clients different from a Web-Browser aren't always compatible with redirects. If the previous point is solved via a redirct, there has to be a flag to prevent a redirect to keep the compatibility with such (automated) clients.

Lifecycle

This section explains the touch-points of the existing JSF Request Processing Lifecycle and the WindowId API.

- When is the windowId created? Depends on the algorithm selected to handle the windowId.
  - Short answer: as soon as possible. At least before the JSF Request Processing Lifecycle starts.
  - In http://wiki.apache.org/myfaces/Drafts/WindowId you can see different solutions. In the solution used in MyFaces Codi when the case is detected where a windowId needs to be created, the faces server responds to the request by sending down a special page, containing only JavaScript, that will ask for the same page as before, but with the windowId. Note that the obvious solution of a 302 redirect is not appropriate here because we want to give the client the responsibility to create the window ID.
  - In other solutions it is generated by the server. So, the api is thought with the intention to provide such details as implementation details just overriding Window object. Look the part on this wiki that says something about different modes (url-Mode, client-Mode).
- When is it updated?
  - It is updated when the application triggers the creation of a new window. Here it also depends on the algorithm selected to handle the windowId. Each strategy to handle windowId has its flaws, and there is no perfect solution for it, because it is a failure of the underlying protocol (http).
- How is it stored during the execution of the lifecycle?
  - Since the windowId does not change at any moment over the view lifetime **within** the **same** browser window, it can be cached in the request scope. Two views can receive the same windowId if and only if the views are rendered on the same browser window, which happens in case of any std. JSF navigation or when for example a GET occur in the same browser window.
- Who renders the windowId to the response, and when?
  - In the proposal, the ResponseStateManager should render the windowId in a hidden field as a **fallback** e.g. in case of URL rewriting. The windowId stored in URLs and hidden field/s have to be in sync. The api proposed looks just like Flash object, but one idea is use Window object to fix Flash scope. If there is a windowId identifier, it can be used for Flash object.

This proposal doesn't propose a new Scope. It just proposes an id for identifying a browser-tab/window.

Suggested APIs

ExternalContext (and ExternalContextWrapper)

String #getWindowId

#setWindowId(String)

Window #getWindow

javax.faces.context.Window

#calculateWindowId(FacesContext)

Extract the windowId from the current request

#createWindowId(FacesContext)

Creates a new Window-Id. Since it might be used e.g. for pop-ups it shouldn't call #setWindowId automatically.

Further discussions needed for

#doPrePhaseActions(FacesContext)

#doPostPhaseActions(FacesContext)

#encodeXYZ

ResponseStateManager

WINDOW_ID_PARAM

//Hidden input field name to store the windowId for POST requests.
public static final String WINDOW_ID_PARAM = "javax.faces.WindowId";

Suggested Rules

Restoring the Window-Id

The window-id gets restored before the JSF Request-Lifecycle starts (e.g. directly after restoring the Flash Scope).

Levels (javax.faces.WINDOW_ID_MODE)

- none
- url
- custom

none-Mode

By default Window-Id's are deactivated to ensure backward compatibility.

url-Mode

That's a very simple approach which has some disadvantages. It just adds the Window-ID to all URLs. That works if users don't open e.g. a link in a new tab (it would clone a window - that isn't nice but not worse than the HTTP session itself). Furthermore, a "drop" script detects the "open in new tab" use-case and drops the current window-id and requests the page again with a new window-id.
There must be a per-use way to disable the inclusion of the URL in a link.

custom-Mode

This mode allows that a JSF implementation or RenderKit provides a proprietary mode which is even more optimized - e.g. using an intermediate page which can be stored in the local-storage of HTML5.
Values: custom:name of the mode e.g.: <param-value>custom:intermediate-page</param-value>

Open Topics

Also is it better to have new window-ids per request, if no js is available, or one window-id for all windows?

Concrete rules for a Request-Token

Max. window-count

Factory for the window

WindowWrapper

Window#close

JS-API

jsf.getWindowId()

returns the current window id of the current window/dialog.
The windowId must be present in the url otherwise a null is returned.

Changes in the behavior on the ajax side

On the ajax side the parameter javax.faces.WindowId must be passed down. If the parameter is stored
in a hidden field like it is proposed before then this happens automatically.
If not then the jsf.ajax.request must add this value before submitting the form, if present on the window url.

Under discussion, additional methods needed?

Any api needed for handling the open in new tab usecase, http get usecase, open or close windows?

Handling of Pop-ups and Dialogs

Rules for automated entry points (to avoid that new windows get created and are around for a long time)

Topics for JSF 2.3+

Portlets

Keepalive (the frequency should decrease over time to allow proper cleanup in case of max. window-count)

Optional onunload hook (in some browsers it just works with a sync. request - and not with async. requests)

Summary of the Discussion in the EG

1 Suggestions based on the status in CODI:

1.1 General Definitions regarding Window-ID
---
a) a window-id always refers to the tab to allow a proper isolation (it should/can be stored in the browsers window.name)

b) a window-id always is unique in the session of the user. A session contains exactly 1 window-id per browser tab.

c) rendering multiple window-ids per browser tab is not allowed hence also not in portlet mode.(because the window-id is stored in the browsers window. name)

1.2 Modes and special cases
---
1) no mode: no window-id is generated - legacy mode

2) url-Mode: the window-id is stored in the browser url
hidden fields are shadowed into the forms also to keep the window-id support for post requests.
windowId parameters are added to http get links.
Server side the windowId needs to be processed over the sent windowId parameters.

This is even the same for Portlet cases since the window-id has
a referential pattern of 1:n to the scopes
The same goes for sub scopes.

If a new tab is opened a Javascript script detects that use case and basically re-renders the page with a new window-id dedicated to the tab.
How that is done is up to the implementation. The basic implementation is to check for an empty or non-matching window.name and then redirect the page without a window-id in the url to force a new one.

3) custom-Mode:
Allows to use a custom mode provided by the JSF implementation or a custom RenderKit.

```
Value: custom:[name of the mode]

e.g.:
<param-value>custom:intermediate-page</param-value>
```

TBD

5) Bookmark case or a href … target=_new: This is handled in CODI by opening a new window-id or recycling the old one (bookmarking case with same windowid in the bookmark as the existing one)

2. Status quo of the spec discussion compared to CODI:
---

a) URL mode: checked is discussed.

b) overriding the URL not yet fully discussed: since it opens another can of worms (f)

c) synchronizing the URL with hidden fields: checked is discussed due to having hidden fields holding javay.faces.windowId

d) Open in new tab script: soon under discussion

e) hidden field only mode: Breaks e.g. the get requests and browser refresh, therefore not doable

f) jsf.getWindowId can be simplified (is now in the proposal) if (b) is omitted or postponed, otherwise it needs to be investigated, if the current implementation can hold. The entire code was programmed under the assumption that we have a window-id per Portlet use case, which cannot hold for (d) see also (g)

g) Ajax protocol ViewRoot either redundant, or in case of (b), cannot work on ViewRoot level
because single forms can if override is enabled be updated with different ViewRoots.
(my mistake i was under the wrong assumption between window-id and scope and because of (g))

h) window-id per Portlet, is there a use case for that one? Especially since it breaks the new
tab detection jsf.getWindowId can hold for (g)

i) server side part of the window-id handling, where to be rendered, under discussion

h) open point: what to do with the a hrefs and pages where you don´t want to have a window id. Should a link to a different window-id be possible, how can you turn it off/on.