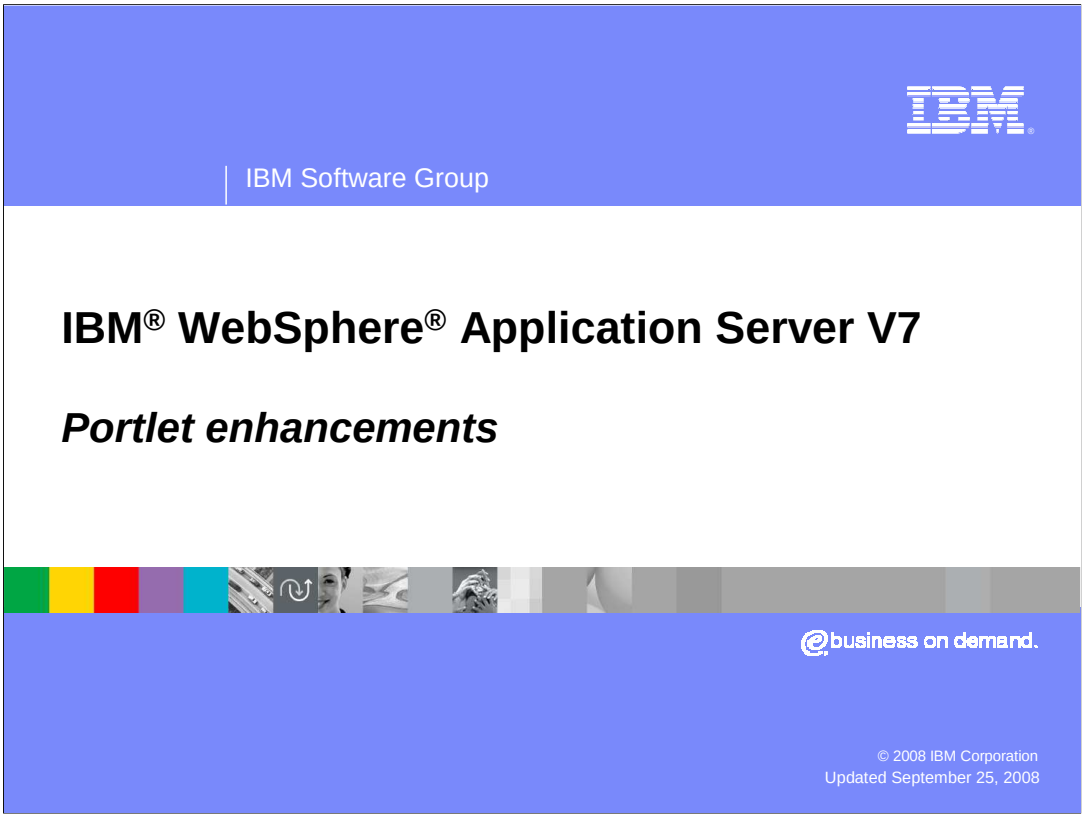


This presentation will discuss enhancements to the portlet support in WebSphere Application Server version 7.

Agenda

- Portlet history
- New features in JSR 286

First this presentation will explain the past portlet support offered by WebSphere Application Server, then it will explain new feature and support for JSR 286, or the portlet 2.0 specification.

Section

Portlet history

This section will discuss the portlet support in past versions of WebSphere Application Server.

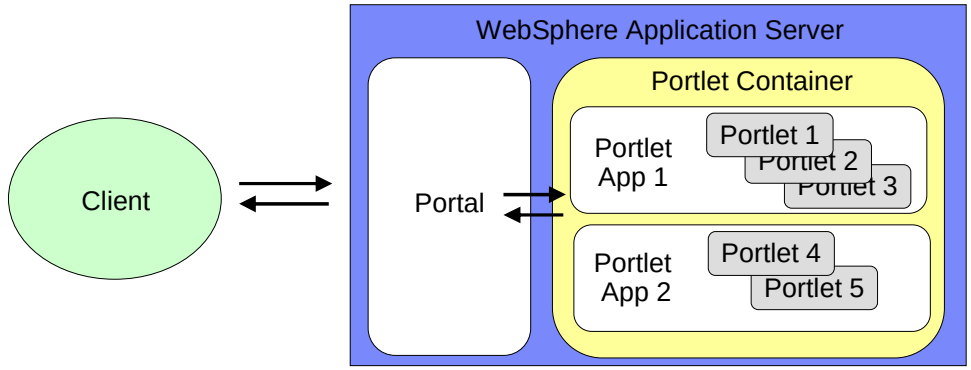
Portlet support overview

- WebSphere Application Server V6.1 introduced support for running JSR 168 compliant portlets
  - Works with the Web container to provide a runtime environment for JSR 168 portlets
  - Portlet container provides portlet runtime environment and life cycle management
  - Did not include advanced capabilities of WebSphere Portal, such as portlet aggregation, personalization, and collaboration

WebSphere Application Server V6.1 provided support for running JSR 168 compliant Portlet applications. The Portlet container is an extension of the Web container, and provides life cycle management and runtime services for Portlets. It allows Portlets to be called directly from a Web browser by URI, and enables Portlet markup to be included in the output of Servlets or Java™ Server Pages. While WebSphere Application Server provides a Portlet runtime environment, it does not replace the WebSphere Portal product, and does not provide the advanced capabilities of that product. WebSphere Portal Server capabilities include portlet aggregation and page layout, personalization and member services, and collaboration features.

Portlet architecture

- Portlet API and container
- Defined a portlet application



The portlet architecture that was developed based on JSR 168, defined an API for developing portlets and created a portlet container. The specification also described portlet applications, and how they can be built up from individual portlets. It also describes the relationship between a portal and actual portlet applications, and how a client communicates with a portal.

Portlet application growth

- JSR 168 defined basic portlet capabilities
- Solved basic use cases and scenarios
- Need for more advanced features
  - Portlet aggregation and collaboration
  - Coordination beyond the application session scope

JSR 168, the first version of the portlet specification, dealt with more basic use cases and scenarios. It defined the basic portlet capabilities needed to create a portlet application, but was not overly focused on advanced functionality. As portlet developers became more familiar with the specification, more advanced features were needed, like portlet aggregation and collaboration, and coordination between portlets beyond the application session data. Some of the need for advanced functionality was added in production Like IBM Portal Server, some have needed a revision of the portlet specification. This has led to the new portlet 2.0 specification, or JSR 286.

Section

JSR 286

The next section will discuss the details of JSR 286, or the portlet 2.0 specification.

Additions in JSR 286

- Improved coordination
  - Events and shared parameters between portlets
- Better support for frameworks using portlets
  - Struts, Spring, and others
- Improvements to serving resources with portlets
- Ability to set cookies or HTTP headers
- Shared cache entries
  - Response can be cached across users

The portlet 2.0 specification contains many additions, including improved coordination between portlets, using events and shared parameters, better ability to develop portlet applications using frameworks such as Struts, Spring, and others. The specification also includes improvements to serving resources from within portlets, and the ability to set cookies or HTTP headers with the portlet API. And finally, shared cache entries for portlet applications.

Coordination in JSR 286

- JSR 286 allows portlets to share session data across Web application boundaries
  - Allows portlets to gather information from other portlets outside a specific application
- Data can be scoped to the current user session
- JSR 286 adds two more mechanisms for information exchange
  - Shared navigational state – portal-managed URL information that is accessible to multiple portlets
  - Event-based communication – send and receive events with a publish-subscribe model

JSR 286 helps provide for improved coordination between portlets. Based on the updated specification portlets can now share session data across Web application boundaries. This allows portlets to gather information from other portlets outside the same application. Data can also be scoped to the current user session.

JSR 286 adds two more mechanisms for information exchange. Shared navigational state allows portal-managed URL information that is accessible to multiple portlets and Event-based communication which allows portlets to send and receive events with a publish-subscribe model controlled by the portal.

Shared navigational state

- No new programming concept for JSR 286 portlets
  - Reading and writing render parameters are already part of JSR 168 – shares this information between portlets
- Simple and powerful concept
  - Typical use cases for coordinating portlet view state can be solved with this model
  - For example, navigator and content display portlet share the same parameter “content ID” → automatic synchronization
- Only declarative extensions necessary
  - Portlets declare certain render parameters as public
  - It is up to the portal how this information is evaluated to actually share values between portlets

The shared navigational state is not a new programming model for portlets. Reading and writing render parameters was already part of JSR 168; this expands that by sharing the information between portlets. It is a simple and yet very useful concept, allowing for use cases such as coordinating the portlet view state by maintaining a shared parameter that can be used by both a navigator portlet and a content display portlet to synchronize their states. This is all done declaratively, with specific portlets specifying which render parameters will be made public. It is then up to the portal itself to decide how to evaluate and share this data between portlets.

Code sample: public render parameters

- Deployment descriptor only
  - Declare a “local” parameter identifier and a “global” name that is common between portlets

```
<public-render-parameter>
   <identifier>zip</identifier>
   <name xmlns:x=”http://acme.com/params”>
      x:address.zipcode
   </name>
</public-render-parameter>
<portlet>
   <portlet-name>portletA</portlet-name>
   …
   <supported-public-render-parameter>zip
   </supported-public-render-parameter>
</portlet>
```


This shows a simple example of how to declare a shared parameter in the deployment descriptor. A local parameter identifier is declared and a global name is used that will be common between all the portlets that use this parameter. So in this case the zip code parameter is identified as zip, and then listed as a supported public render parameter.

Shared navigational state

- Advantages of using public render parameters
  - Provides bookmarkability and back button support
  - Less processing overhead
  - Parallel rendering of portlets possible
- Portlet coordination is limited to synchronizing simple view state information
  - Only defines new view state, no server side state changes (HTTP GET semantics)
  - No active notification that something has changed
  - Only (short) strings as data types
- Portlets that want to transfer complex data or react to messages with code need a more powerful concept: ***events***

Using public render parameters offers several advantages. It can be used to provide bookmark capabilities and support for a browsers back button. It uses less processing overhead, and allows for parallel rendering of portlets, which both provide for better performance. Portlet coordination is then limited to synchronizing the simple view state information. This simplicity means that only short strings can be used as data type, and there is no active notification when something changes. Portlet applications that require support for more complex data types or that need a more dynamic notification model have an alternate method, called events.

Events

- JSR 286 introduces a loosely coupled event model for portlets
  - Similar to servlet filters
    - Can be chained
  - Portlets can generate or receive events
  - A portal is needed to manage and distribute events
- Event handling is an additional step during the action phase
  - Allows state changes to occur
  - Rendering cannot occur until event handling is finished

JSR 286 introduces a loosely coupled event model that can be used with portlets. It works similar to servlet filters and events can be chained together. Portlets can generate or receive events but a portal is needed to manage and distribute these events. Handling the events is an additional step during the action phase; this allows state changes to occur. But is also means rendering cannot occur until event handling is finished.

Events

- WebSphere Application Server version 7 supports portlet events as a programming model
  - A portal is needed to use this programming model and transport events
- Event distribution is not part of the standard
  - Portlets never call each other, control always flows through the portal
  - Different portals may use different ways to match published events to potential receivers

WebSphere Application Server version 7 supports portlet events as a programming model, but a portal is needed to use this programming model and transport events.

Event distribution is not part of the standard, portlets never call each other, control always flows through the portal. Because it is not standardized different portals may use different ways to match published events to potential receivers. So, developers should be cautious when using this new programming model and investigate the event handling providing by the specific portals used.

Improved framework support

- JSR 286 adds features to better support the use of Web frameworks in portlet applications
- Allow easy bridging from portlet to servlet
  - Allows servlet dispatching from processAction, processEvent, render or serverResource
- Can optionally provide a portlet-scoped session to servlets
- Extended JSP tag library

JSR 286 also adds features to better support the use of Web frameworks in portlet applications. In concept this should allow for easy bridging from portlet to servlet and allows servlet dispatching from processAction, processEvent, render or serverResource. Developers can optionally provide a portlet-scoped session to servlets. There is also an extended JSP tag library that better support frameworks.

Improved resource serving

- With JSR 168, portlets can only output fragments for a portal page
  - Resources like images can be packaged in the WAR file
  - Dynamically generated resources can be served by a separate servlet packaged in the portlet WAR file
- JSR 286 supports resource serving in the portlet
  - Portlet has full control of the output stream
  - Portlet context available (render params, portlet mode, window state, preferences, ...)
  - Can serve binary content such as PDF, XML output or HTML for pop-up windows without portal framework

With JSR 168, portlets can only output markup fragments for a portal page, this limits their ability to server other types of resources such as PDFs or XML. Resources like images can be packaged in the WAR file and dynamically generated resources can be served by a separate servlet packaged in the portlet WAR file, but these are more limiting for developers.

JSR 286 supports resource serving from within the portlet. It allows the portlet full control of the output stream, and the ability to provide the portlet context. Portlets can than serve binary content such as PDF, XML output or HTML for pop-up windows without a portal framework.

Improved resource serving

- Programming model
  - New ResourceURLs that trigger a new life cycle method serveResource
  - No state changes on navigational state (render params, portlet mode, window state) or shared state allowed
  - Protected using the portal access control
- Different cache levels of resource URLs
  - For supporting caching of the resource at the browser
  - Three types introduced: FULL, PORTLET, PAGE
- Resource Ids
  - Set a specific resource IDs on a resource URL
  - Default behavior of GenericPortlet is to try to forward the resource serving to the resource IDs specified

The programming model for resource serving has new ResourceURLs that trigger a new life cycle method serveResource. It does not allow state changes on navigational state (render params, portlet mode, window state) or shared state, and is protected using the portal access control. Different cache levels of resource URLs are provided for supporting caching of the resource at the browser. Developers can set a specific resource IDs on a resource URL.

Cookies and HTTP headers

- Portlets in JSR 186 are not able to set cookies and HTTP headers
  - Portal has the control over the output stream to the client and body content may already be written
- Portlets in JSR 286 can set cookies and HTTP headers
  - In all life cycle methods
  - Also available for render response
    - But may be overridden by the portal or other portlets
  - Restrictions on cookies
    - Cookies may be stored on the portal or get re-written and thus not accessible on the client

Portlets in JSR 186 can not set cookies and HTTP headers, and developers found this to be rather limiting for Web applications that used portlets. The reason behind this was that the portal has the control over the output stream to the client and body content may already be written. Now, portlets in JSR 286 can set cookies and HTTP headers. This is possible in all life cycle methods. It is also available on the render response, but may be overridden by the portal or other portlets. There are some restrictions on cookies, they may be stored on the portal or get re-written and thus not accessible on the client.

Setting cookies and HTTP headers

- Examples of the API usage
- HTTP headers
  - Setting: with the set / add property methods on the response
  - Retrieving: with the getProperty methods on the request
- Cookies
  - Setting: addProperty(javax.servlet.http.Cookie)
  - Retrieving: javax.servlet.http.Cookie[] getCookies()

This slide shows some examples of how to set and retrieve HTTP headers and cookies using the new API from the portlet 2.0 specification.

Section

Summary

Following is the summary for the presentation.

Summary

- JSR 286 provides key improvements in the portlet programming model
  - Events and shared parameters between portlets
  - Improved coordination with frameworks
  - Better serving of resources
- WebSphere Application Server V7 provides support for JSR 286 based portlets

In summary, JSR 286 provides key improvements to the portlet programming model. It allows for events and shared parameters between portlets, improved coordination with frameworks, better serving of resources, and using cookies. WebSphere Application Server V7 provides support for the JSR 286 portlet programming model.

Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_WASv7_PortletEnhancements.ppt

This module is also available in PDF format at: ../WASv7_PortletEnhancements.pdf

You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, copyrights, and disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM WebSphere

A current list of other IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

Java, JSP, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements or changes in the products or programs described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (for example, IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.