Tổng quan

Portal – Liferay Portal

Vportal Team
Công ty VietSoftware
Địa chỉ: Tầng 8, số 51 Lê Đại Hành, Hà Nội, Việt Nam
Điện thoại: (84-4) 9745699; Fax: (84-4) 9745700
E-mail: **contact@vietsoftware.com**
Web site: **http://www.vietsoftware.com**

10/2007

Khái niệm Portal

Định nghĩa một cách tương đối, Portal là một phần mềm ứng dụng Web-based, định danh và xác thực người dùng đăng nhập, từ đó sẽ cung cấp giao diện mang tính cá nhân hóa cho người sử dụng. Thông qua giao diện này, người dùng dễ dàng truy cập, khai thác, tìm kiếm, giao tiếp với các ứng dụng, các thông tin, và với những người dùng khác.

Đứng trên khía cạnh công nghệ, ngày nay portal được coi như một giải pháp(frame work) mà thông qua đó chúng ta có thể quy tụ, quản lý nhiều nguồn thông tin(bao gồm thông tin và ứng dụng phần mềm) khác nhau vào trong một thực thể phần mềm duy nhất-phần mềm portal. Từ đó, các thông tin được phân phối dưới dạng các dịch vụ cho từng người dùng khác nhau tuỳ thuộc vào nhóm quyền, vào nhu cầu cũng như mục đích của người dùng đó.

Có thể nói, Portal như một cổng vào vạn năng cho người dùng tìm kiếm thông tin và tác nghiệp một cách thuận lợi và dễ dàng.

Liferay Portal là một phần mềm nguồn mở được phát triển bởi công ty Liferay. Đây là một portal nguồn mở được đánh giá cao nhất về tất cả các mặt trong cộng đồng nguồn mở hiện nay(Xem chi tiết trong phần "Đánh giá các portal").

So sánh portal với website thông thường

Sang Tiếng Việt, Web Portal được dịch là "Cổng giao tiếp điện tử", "Cổng giao dịch điện tử" hoặc ngắn gọn hơn: "Cổng điện tử". Tuy nhiên, cũng như tên tiếng Anh của chúng, các từ này thật sự chưa thể phản ánh hết được chính xác thế nào là một Portal. Để làm rõ bản chất của Portal chúng ta đưa ra các so sánh giữa Portal với một Website thông thường sau đây.

+ Khả năng đăng nhập một lần tới tất cả các tài nguyên được liên kết với Portal. Nghĩa là, người dùng chỉ cần một lần đăng nhập là có thể vào và sử dụng tất cả các ứng dụng đã được tích hợp trong Portal đó mà người dùng này có quyền. Một website thông thường không có được khả năng đăng nhập một lần.

+ Khả năng cá nhân hóa theo người sử dụng.

Đây là một trong những khả năng quan trọng của Portal, giúp nó phân biệt với một website thông thường. Portal cá nhân hóa nội dung hiển thị, thông

thường đây là sự lựa chọn một cách tự động dựa trên các quy tắc tác nghiệp, chẳng hạn như vai trò của người sử dụng trong một tổ chức. Ví dụ khi một người mua hàng đăng nhập vào hệ thống, Portal sẽ hiện ra một danh sách các sản phẩm mới. Hoặc nếu cần quan tâm đến các lĩnh vực khảo cổ thì Portal có thể cung cấp các thông tin bảng danh sách các đồ cổ.

Trong khi đó, website thường không hỗ trợ, nếu có chỉ ở mức độ rất nhỏ, không phải là đặc điểm nổi bật.

+ Khả năng tùy biến.

Đây là một khả năng tiêu biểu của một Portal.

Ví dụ một giao diện Portal có mục thông tin thời tiết, chúng ta có thể bỏ phần thông tin này đi nếu chúng ta không quan tâm đến nó. Hoặc chúng ta có thể thay đổi cách hiển thị của Portal. Ví dụ như thay vì hiển thị bằng font chữ màu xác định chúng ta có thể thay nó bằng chữ màu đỏ, hay có thể tự thay đổi giao diện của Portal nếu mặc định chức năng A được đặt sau chức năng B, nếu không thích chúng ta có thể thay đổi lại thứ tự hiển thị này. Đặc tính này tương tự như màn hình desktop của chúng ta.

Một vài website thường có nhưng chỉ dừng lại ở mức độ dựng sẵn, người dùng chỉ có thể lựa chọn một vài giao diện đã có, mà không tự mình thay đổi từng mục một cách tùy ý.

+ Liên kết truy cập tới hàng trăm kiểu dữ liệu, kho dữ liệu, kể cả dữ liệu tổng hợp hay đã phân loại.

Portal nó có khả năng liên kết tới tài nguyên dữ liệu rộng lớn, gồm nhiều kiểu dữ liệu từ dữ liệu thông thường đến siêu dữ liệu.

Website thường chỉ sử dụng các liên kết để tới các site khác nhưng nội dung chủ yếu vẫn chỉ tập trung trong trang đó.

+ Khả năng liên kết và hợp tác người dùng(hỗ trợ rất tốt).

Portal không chỉ liên kết chúng ta với những gì chúng ta cần mà còn liên kết với những người mà chúng ta cần. Khả năng liên kết này được thực hiện bởi các dịch vụ hợp tác. Chẳng hạn, đơn giản như khả năng hỗ trợ tương tác hai chiều giữa công dân và cơ quan chức năng trong việc hỏi đáp chính sách, pháp luật, hay cao hơn nữa là portal có khả năng trợ giúp xây dựng các cuộc giao lưu, tổ chức hội thảo trực tuyến hay tạo ra một cộng đồng ảo.

Website thông thường không hỗ trợ tính năng này.

Các tính năng cơ bản

Tuy có nhiều loại cổng thông tin tích hợp, cung cấp nhiều loại dịch vụ và ứng dụng khác nhau, nhưng tất cả các loại cổng thông tin tích hợp đều có chung một số tính năng. Các tính năng này là được sử dụng như một tiêu chuẩn để phân biệt giữa cổng thông tin điện tử tích hợp với một Web site hoặc một ứng dụng chạy trên nền tảng Web. Các tính năng đó bao gồm:

1. **Khả năng cá nhân hoá (Customization hay Personalization):** cho phép thiết đặt các thông tin khác nhau cho các loại đối tượng sử dụng khác nhau theo yêu cầu. Tính năng này dựa trên hoạt động thu thập thông tin về người dùng và cộng đồng người dùng, từ đó cung cấp các thông tin chính xác tại thời điểm được yêu cầu.
2. **Tích hợp và liên kết nhiều loại thông tin (Content aggregation):** cho phép xây dựng nội dung thông tin từ nhiều nguồn khác nhau cho nhiều đối tượng sử dụng. Sự khác biệt giữa các nội dung thông tin sẽ được xác định qua các ngữ cảnh hoạt động của người dùng (user-specific context), ví dụ như đối với từng đối tượng sử dụng sau khi thông qua quá trình xác thực thì sẽ được cung cấp các thông tin khác nhau, hoặc nội dung thông tin sẽ được cung cấp khác nhau trong quá trình cá nhân hoá thông tin.
3. **Xuất bản thông tin (Content syndication):** thu thập thông tin từ nhiều nguồn khác nhau, cung cấp cho người dùng thông qua các phương pháp hoặc giao thức (protocol) một cách thích hợp. Một hệ thống xuất bản thông tin chuyên nghiệp phải có khả năng xuất bản thông tin với các định dạng đã được quy chuẩn, ví dụ như RDF (Resource Description Format), RSS (Rich Site Summary), NITF (News Industry Text Format) và NewsXML. Ngoài ra, các tiêu chuẩn dựa trên XML cũng phải được áp dụng để quản trị và hiển thị nội dung một cách thống nhất, xuyên suốt trong quá trình xuất bản thông tin. Các tiêu chuẩn dựa trên XML này cho phép đưa ra giải pháp nhanh nhất để khai thác và sử dụng thông tin trên các Web site khác nhau thông qua quá trình thu thập và bóc tách thông tin với các định dạng đã được quy chuẩn.
4. **Hỗ trợ nhiều môi trường hiển thị thông tin (Multidevice support):** cho phép hiển thị cùng một nội dung thông tin trên nhiều loại thiết bị khác nhau như: màn hình máy tính (PC), thiết bị di động (Mobile phone, Wireless phone, PDA), sử dụng để in hay cho bản fax…. một cách tự động bằng cách xác định thiết bị hiển thị thông qua các thuộc tính khác nhau. Ví dụ: cùng một nội dung đó, khi hiển thị trên màn hình

máy tính thì sử dụng HTML, nhưng khi hệ thống xác định được thiết hiển thị là PDA hay mobile phone, hệ thống sẽ loại bỏ các ảnh có trong nội dung và tự động chuyển nội dung đó sang định dạng WML (Wireless Markup Language) để phù hợp cho việc hiển thị trên màn hình của thiết bị di động.

5. **Khả năng đăng nhập một lần (Single Sign On):** cho phép dịch vụ xuất bản thông tin hoặc các dịch vụ khác của portal lấy thông tin về người dùng khi hoạt động mà không phải yêu cầu người dùng phải đăng nhập lại mỗi khi có yêu cầu. Đây là một tính năng rất quan trọng vì các ứng dụng và dịch vụ trong portal sẽ phát triển một cách nhanh chóng khi xuất hiện nhu cầu, mà các ứng dụng và dịch vụ này tất yếu sẽ có các nhu cầu về xác thực hoặc truy xuất thông tin người dùng.
6. **Quản trị portal (Portal administration):** xác định cách thức hiển thị thông tin cho người dùng cuối. Tính năng này không chỉ đơn giản là thiết lập các giao diện người dùng với các chi tiết đồ hoạ (look-and-feel), với tính năng này, người quản trị phải định nghĩa được các thành phần thông tin, các kênh tương tác với người sử dụng cuối, định nghĩa nhóm người dùng cùng với các quyền truy cập và sử dụng thông tin khác nhau.
7. **Quản trị người dùng (Portal user management):** cung cấp các khả năng quản trị người dùng cuối, tuỳ thuộc vào đối tượng sử dụng của portal. Tại đây, người sử dụng có thể tự đăng ký trở thành thành viên tại một công thông tin công cộng (như Yahoo, MSN…) hoặc được người quản trị tạo lập và gán quyền sử dụng tương ứng đối với các công thông tin doanh nghiệp. Mặt khác, tuỳ vào từng kiểu portal mà số lượng thành viên có thể từ vài nghìn tới hàng triệu. Hiện tại phương pháp phân quyền sử dụng dựa trên vai trò (Role-based security) được sử dụng như một tiêu chuẩn trong các hoạt động xác định quyền truy cập và cung cấp thông tin cho các đối tượng khác nhau trong các portal cũng như các ứng dụng Web.

Lợi ích của hệ thống Portal

Hệ thống Portal hỗ trợ cộng đồng người dùng trực tuyến, các cán bộ, nhân viên, các đối tác và các nhà cung cấp... dưới nhiều hình thức kết hợp khác nhau. Cơ sở hạ tầng Portal giúp việc khởi tạo, tích hợp, quản lí và cá nhân hóa toàn diện các thông tin và ứng dụng cho mỗi người dùng riêng biệt phục vụ các nhu cầu và sở thích của một cộng đồng riêng biệt.

Người dùng có thể dễ dàng hình dung được các dịch vụ mà hệ thống portal cung cấp như sau:

1. Các dịch vụ cơ bản: Post bài định dạng HTML/Document, Danh sách liên kết, Upload/Download Files, Thao tác ảnh…
2. Các dịch vụ giao tiếp công cộng: Forum, Thông báo, Thăm dò - Bỏ phiếu…
3. Các dịch vụ cung cấp thông tin: Thông báo, Bản tin...
4. Các dịch vụ tìm kiếm: Tìm kiếm, Phân loại …
5. Các dịch vụ trợ giúp người dùng: Thông tin cá nhân, Lịch biểu…
6. Các dịch vụ tác nghiệp: Quản lý nội dung, Hợp tác dự án, Quản lý bán hàng, quản lý nhân sự…

Có thể thấy rằng, các lợi ích thực sự của hệ thống Portal này đem lại nhìn từ khía cạnh hiệu quả ứng dụng thực tế đó là:

- ✓ Nâng cao hiệu quả làm việc cho các cá nhân và tổ chức, đối tác... nhờ truy cập bảo mật, tích hợp tới các thông tin và ứng dụng liên quan, cũng như truy cập tổng thể tới tất cả các cá nhân, thông tin, tổ chức và các nhà cung cấp từ bất kì đâu, bất kì khi nào.
- ✓ Cải thiện các tiến trình hợp tác nhờ luồng thông tin tốt hơn giữa con người và các ứng dụng, và nhờ các môi trường cộng tác giúp giảm thời gian để chuyển đổi thông tin thô thành tri thức.
- ✓ Giảm gánh nặng của việc triển khai và quản lí thông tin và các dịch vụ ứng dụng trong một tổ chức.
- ✓ Duy trì, quản lý, mở rộng, nâng cấp, tái sử dụng dễ dàng, tiết kiệm chi phí đầu tư để xây dựng lại hệ thống.
- ✓ Cho phép các hãng thứ 3 tham gia vào việc cung cấp ứng dụng hệ thống, các dịch vụ trung gian... Khả năng này làm phong phú, đa dạng khả năng úng dụng và triển khai của hệ thống Portal.

Phân loại Portal

Việc phân loại Portal có thể có nhiều cách khác nhau. Nếu căn cứ vào đặc trưng của Portal người ta chia Portal thành các loại như sau :

1. Consumer Portal

Cung cấp nhiều lựa chọn cho việc tìm kiếm, chuyển, E-mail, tự sửa khuôn dạng, lựa chọn tin tức, calendar, quản lý địa chỉ liên hệ, các cuộc hẹn, các

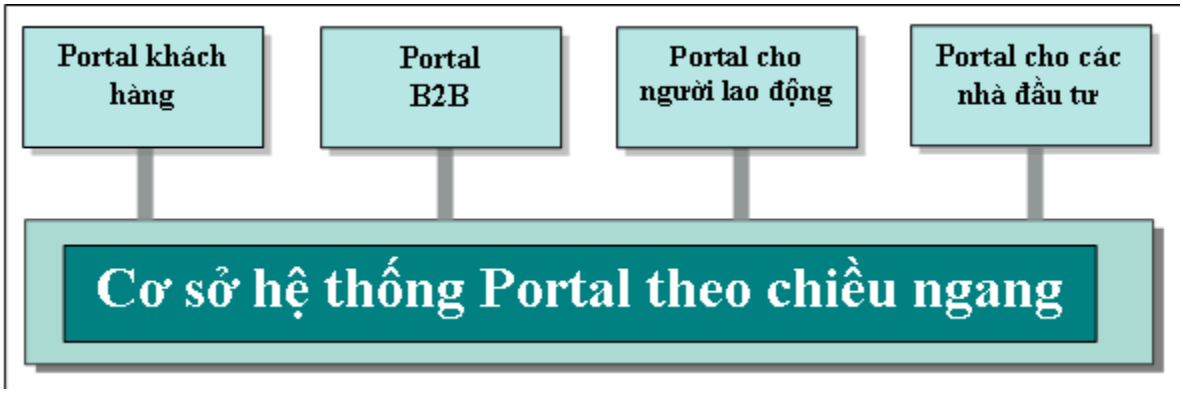
lưu ý, chú thích, các địa chỉ website, real-time chat và các chức năng Intranet, v.v…

2. Vertical Portal

Chuyên cung cấp các thông tin và dịch vụ mang tính chuyên ngành cho một lĩnh vực chuyên môn, khoa học, kinh tế cụ thể nào đó.

3. Horizontal Portal

Nội dung bao trùm nhiều lĩnh vực hoặc nhiều chủ đề trong một lĩnh vực lớn (mang tính diện rộng) như kinh tế, khoa học, công nghệ, y học, thể thao, âm nhạc…, phục vụ các mối quan tâm khác nhau, hỗ trợ bằng các chức năng dịch vụ phong phú, phục vụ cộng đồng, phục vụ tổ chức hành chính.



4. Enterprise Portal

Cung cấp các dịch vụ truy xuất thông tin từ mọi nguồn tài nguyên thông tin trong mạng Intranet của một tổ chức qua một cổng truy cập duy nhất.

5. B2B Portal

Cung cấp các dịch vụ định hướng theo mối quan hệ tương tác thông tin hai chiều giữa các doanh nghiệp (B2B) trong môi trường thương mại điện tử.

6. G2G Portal

Cung cấp các dịch vụ hành chính công theo mối quan hệ tương tác thông tin hai chiều giữa các cơ quan hành chính nhà nước (G2G) trong môi trường trao đổi thông tin điện tử.

So sánh các Portals

1. Các tiêu chí đánh giá

Có một sự khó khăn khi so sánh các Portal vì mỗi Portal trong số chúng dựa trên những yêu cầu khác nhau và các công nghệ khác nhau. Việc so sánh portal dựa trên những tiêu chí đánh giá khác nhau. Những tiêu chí này dựa trên lõi và những yêu cầu lựa chọn từ Portal:

- ✓ Tuân theo JSR-168 (JSR-168 compliant)
- ✓ Tính dễ dàng cài đặt(Ease to installation)
- ✓ Tài liệu chuẩn(Documentation Standard)
- ✓ Hỗ trợ trực tuyến(Online Support)
- ✓ Quản lý Portal (Portal Management)
- ✓ Các tài nguyên Portal(Portlet Resources)
- ✓ Khả năng thực thi và tính linh hoạt(Performance & Scalability)
- ✓ Bảo mật (Security)
- ✓ Công nghệ sử dụng(Technology Used)
- ✓ Các đặc điểm của Portal(Portal Features)
- ✓ Sự phụ thuộc server(Server Dependency)
- ✓ Tuân theo chuẩn WSRP(WSRP standard compliant)

2. Đánh giá các Portal

Việc đánh giá chỉ nằm trong danh sách các Portal phổ biến dưới đây:

- uPortal: theo sự sử dụng rất lớn trong các học viện.
- eXo: theo sự phổ biến
- Liferay: theo sự phổ biến, giao diện người dùng và chức năng lựa chọn
- Stringbeans: theo sự dễ dàng sử dụng

2.1.uPortal

uPortal là một Portal Framework được sử dụng rộng rãi trong các học viện và nó chủ yếu nhằm vào những yêu cầu của các tổ chức này. uPortal là một Portal Framework rất ổn định và đã được ra đời thậm chí trước cả JSR-168 specification, theo đó uPortal đã áp dụng những kỹ thuật ko theo chuẩn được gọi là channel. uPortal mặc dù đã tuân theo JSR-168 nhưng hầu hết những

đặc điểm sẵn có trong uPortal vẫn dựa trên tùy biến và giải pháp đã phát triển với các channel adapter hơn là các portlet nguyên thủy. uPortal hỗ trợ portlet thông qua Pluto Portlet Framework. uPortal cũng là open source Portal Framework hỗ trợ nhiều kiểu portal nhất: từ Java portal đến HTML portal, từ text portal đến XML portal.

2.2.eXo Platform

eXo Platform định nghĩa như một portal và một CMS. Có thể coi, eXo Platform là một open source Portal Framework mạnh mẽ với việc hỗ trợ nhiều công nghệ mới. Khả năng thực thi của eXo Platform tốt nhất với thời gian upload portal nhỏ nhất.

2.3.Stringbeans

Stringbeans Portal được tạo nên là một portlet container tuân theo JSR-168 và một framework cho việc quản trị hữu dụng các portal application.

Stringbeans có nhiều đặc điểm thân thiện với user và developer, đặc biệt đánh giá là có documentation và hỗ trợ trực tuyến tốt nhất trong số các open source Portal Framework.

2.4.Liferay

Liferay Portal Enterprise mang nhiều ý nghĩa lớn hơn là một portal container, mà đi kèm với nó là rất nhiều đặc điểm hữu dụng như Content Management System (CMS), tuân theo WSRP, Single Sign On (SSO), hỗ trợ AOP (Aspect Oriented Programming), và nhiều công nghệ mới nhất khác.

Liferay có một thiết kế kiến trúc rất rõ ràng và linh hoạt dựa trên thực tế tốt nhất của J2EE. Do đó, nó có thể sử dụng tốt các server khác nhau như Tomcat, Jetty, JBoss, JRun, Oracle9iAS, Orion, WebLogic, WebSphere hay công nghệ khác nhau như Struts, Tiles, Spring, EJB , JMS, Java Mail, Web

Service…Như vậy, Liferay là một open source portal container hỗ trợ gần như hầu hết JavaServer open source hay thương mại.

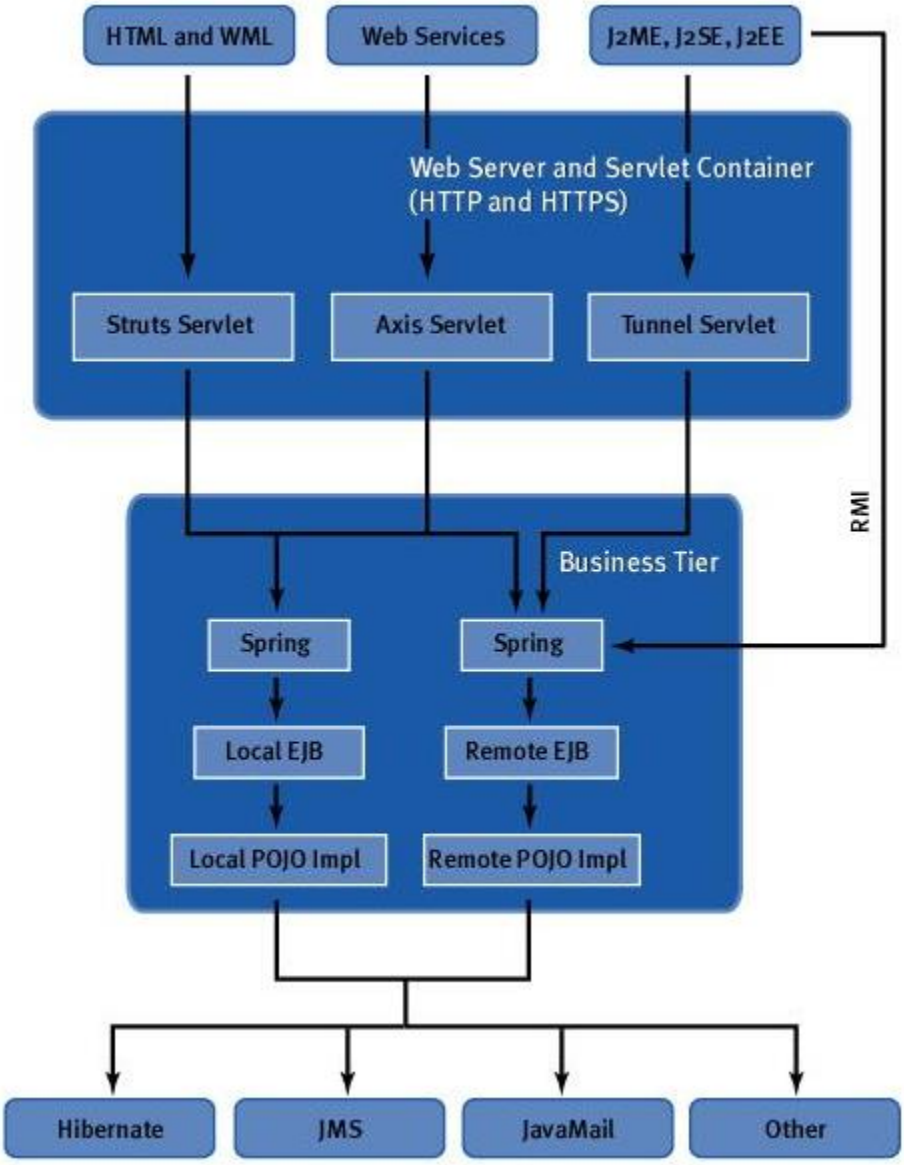
Việc customize các portal page và các portlet trong những open source Portal Framework như eXo Platform là không dễ dàng, và có thể làm rất nhiều trong việc cấu hình, nhưng với Liferay layout management thì rất dễ dàng. Liferay Portal có một GUI dựa trên Web cho phép user tương tác để thiết kế layout của Portal Page mà không cần phải chỉnh sửa bất kỳ file cấu hình nào.

Liferay Portal Enterprise đi kèm với những portlet hữu dụng. Và nếu đem so sánh với các open source Portal Framework khác, Liferay portal có một lượng lớn các portlet tiện ích tuân theo JSR-168 và có thể được sử dụng trong bất kỳ Portal nào chỉ với rất ít thay đổi.

Liferay hỗ trợ WSRP specification cả cho WSRP consumer và WSRP producer như một thực thể của Liferay portal.

Liferay có thể được sử dụng với bất kỳ database nào với chút ít ảnh hưởng tùy theo việc sử dụng Hibernate trong thiết kế của nó. Liferay có các JSP tag lib và nhiều class tiện ích khác trong những package khác nhau để trợ giúp các developer trong việc phát triển portal/portlet.

Ta có xem kiến trúc Liferay portal như hình sau:



Kết quả đánh giá:

Table1: Evaluation Result

Criteria	Portal Framework					
	Sakai 1.5	uPortal	Gridsphere	Exo	Liferay	Stringbeans
JSR-168 Compliance	0	5	5	5	5	5
Ease of Installation	3	5	5	5	5	5
Ease of Use	3	5	4	5	4	5
Documentation	2	2	4	3	3	5
Support Services	3	3	4	4	3	5
Administration of Portal	3	5	4	5	4	5
Customisation	4	3	4	3	5	4
Free Useful Portlets	4	3	4	3	5	3
Performance	2	4	3	4	3	3
Security	3	4	3	4	4	4
Technology Use	3	3	4	5	4	3
Portal Features	2	2	3	5	4	2
Server Dependency	3	3	3	4	5	3
WSRP Compliance	0	3	0	3	3	0
Total	35	49	51	57	58	51

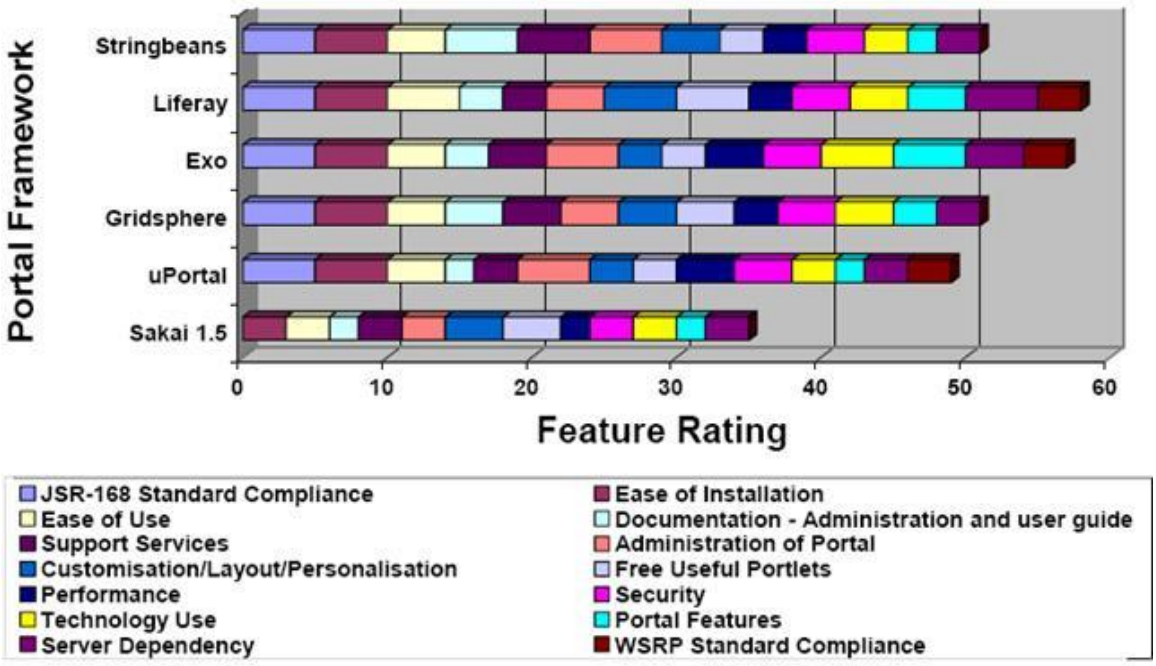


Figure 3: Evaluation Result as Bar Chart

Như vậy, Liferay Portal là một portal nguồn mở được đánh giá cao nhất về tất cả các mặt trong cộng đồng nguồn mở hiện nay.

Kiến trúc, mô hình hoạt động của portal

1.Một số khái niệm

Portal kết hợp các portlet lại với nhau để tạo nên trang portal. Các portlet này được quản lý bởi portlet container. Vậy trước hết ta cần tìm hiểu portal là gì? portlet là gì? portlet container là gì?

Portal là gì?

(Xem phần Khái niệm portal phía trên)

Portlet là gì?

Một portlet là thành phần web dựa trên kỹ thuật Java, được quản lý bởi portlet container, nó xử lý các yêu cầu và sản sinh ra nội dung html động.

Portlet container là gì?

Mọi portlet được triển khai trong một portlet container, portlet container điều khiển toàn bộ chu trình sống của portlet, cung cấp những tài nguyên và môi trường để chạy các portlets đó. Portlet container nhận các yêu cầu từ cổng điện tử và thực thi các yêu cầu trên những portlets được chứa bởi nó.

Portlet container không chịu trách nhiệm kết hợp nội dung sản sinh bởi các portlet. Trách nhiệm đó thuộc về cổng điện tử.

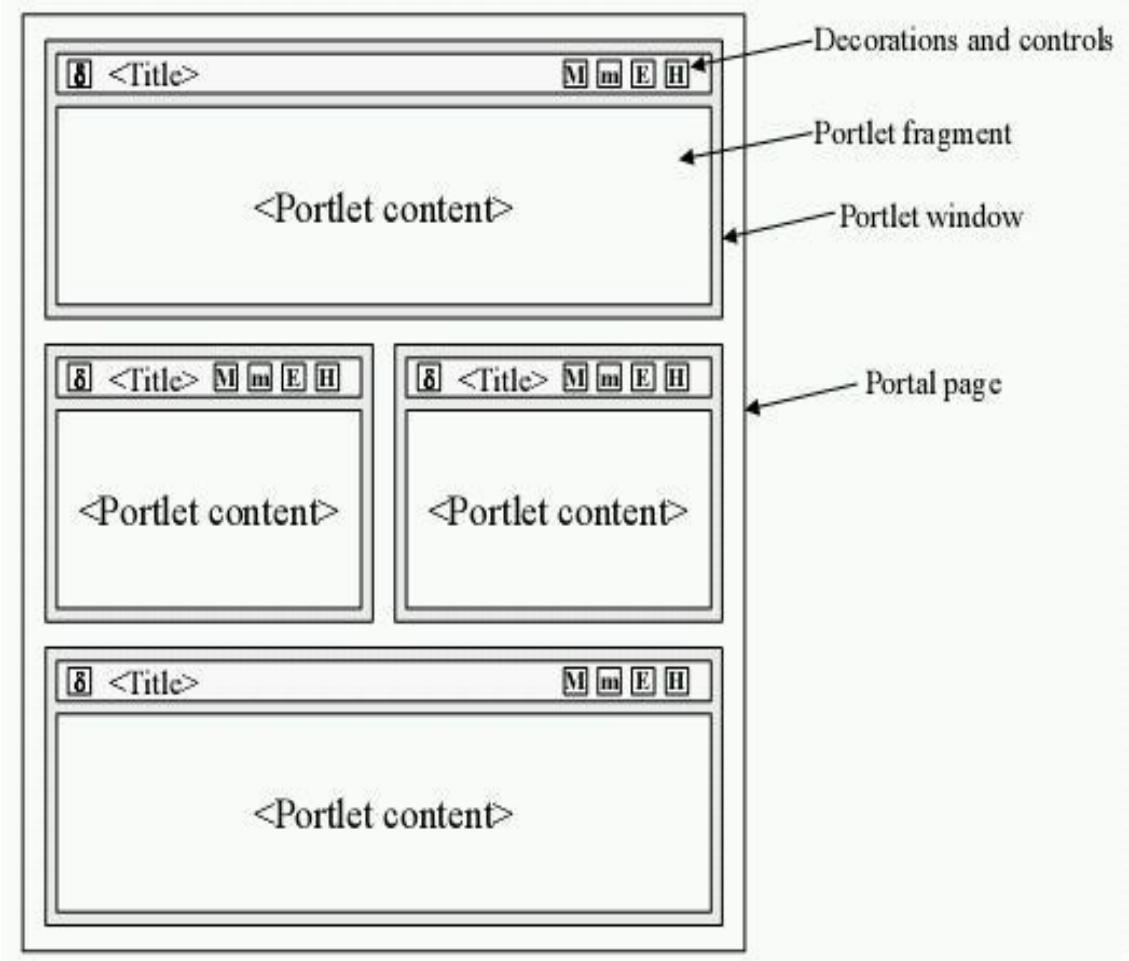
Một cổng điện tử và Portlet container có thể được xây dựng với nhau thành 1 thành phần duy nhất của bộ ứng dụng hay như là 2 thành phần riêng biệt của ứng dụng cổng điện tử.

2. Thành phần của một trang portal

Một trang portal có thể tạo ra một hay nhiều cửa sổ portlet. Mỗi cửa sổ Portlet được tạo bởi hai thành phần : một là decoration, chúng giống như một frame hay window chứa title bar (thanh tiêu đề), controls (những điều khiển), những border của cửa sổ sẽ xuất hiện. Thành phần thứ hai được gọi là portlet fragment, đây là thành phần chứa ứng dụng portlet. Ví dụ trên một trang portal chúng ta có thể thấy vùng thông tin thời tiết, vùng tin tức, vùng giá cả chứng khoán. Nếu chỉ một trong chúng được cập nhật thì những cái còn lại cũng được cập nhật theo. Mỗi vùng thông tin đó là một portlet , bạn có thể thấy chúng có title bar, một vài nút như đóng, phóng to, thu nhỏ, lên xuống,…

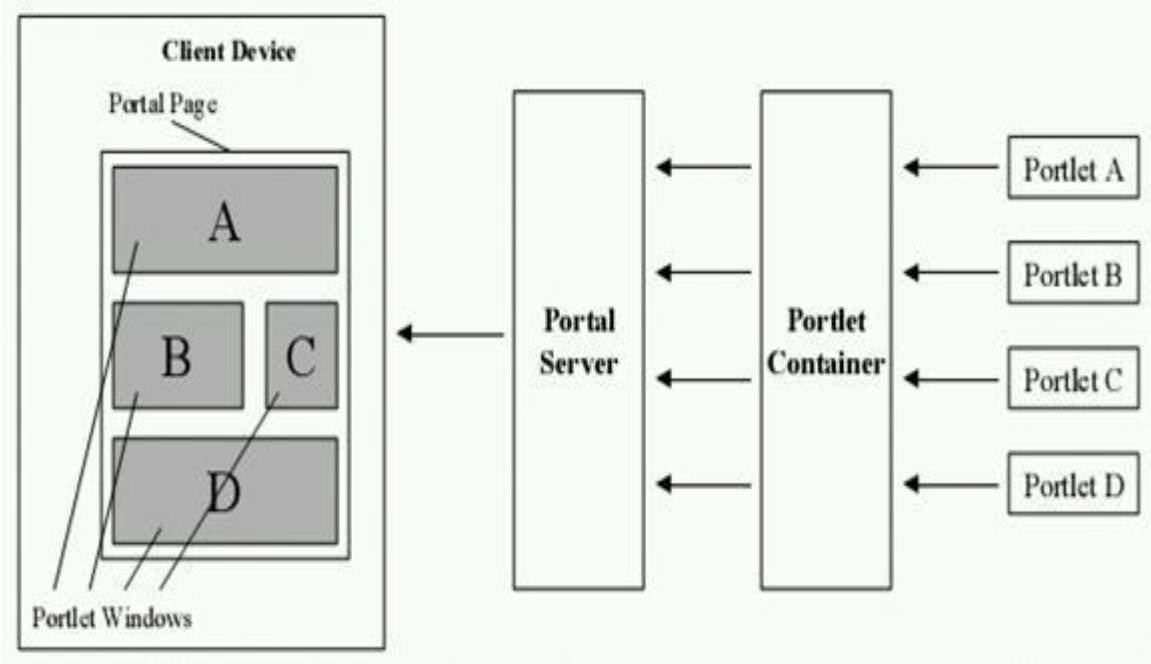
Thực chất, những của sổ này là những ứng dụng khác nhau, được phát triển độc lập. Nhà phát triển tạo ra một portlet như việc tạo một ứng dụng , nén lại dưới file war. Sau đó nhà quản trị portal sẽ cài đặt file war này trên máy chủ và tạo trang.

Bước tiếp theo là người sử dụng có thể cho ứng dụng mà họ muốn dùng vào trong trang của họ. Ví dụ, nếu một người sử dụng không thích cập nhật những thông tin chứng khoán và anh ta lại muốn cập nhật tin tức thể thao. Khi đó anh ta có thể thay portlet chứng khoán bằng một portlet cập nhật tin thể thao.



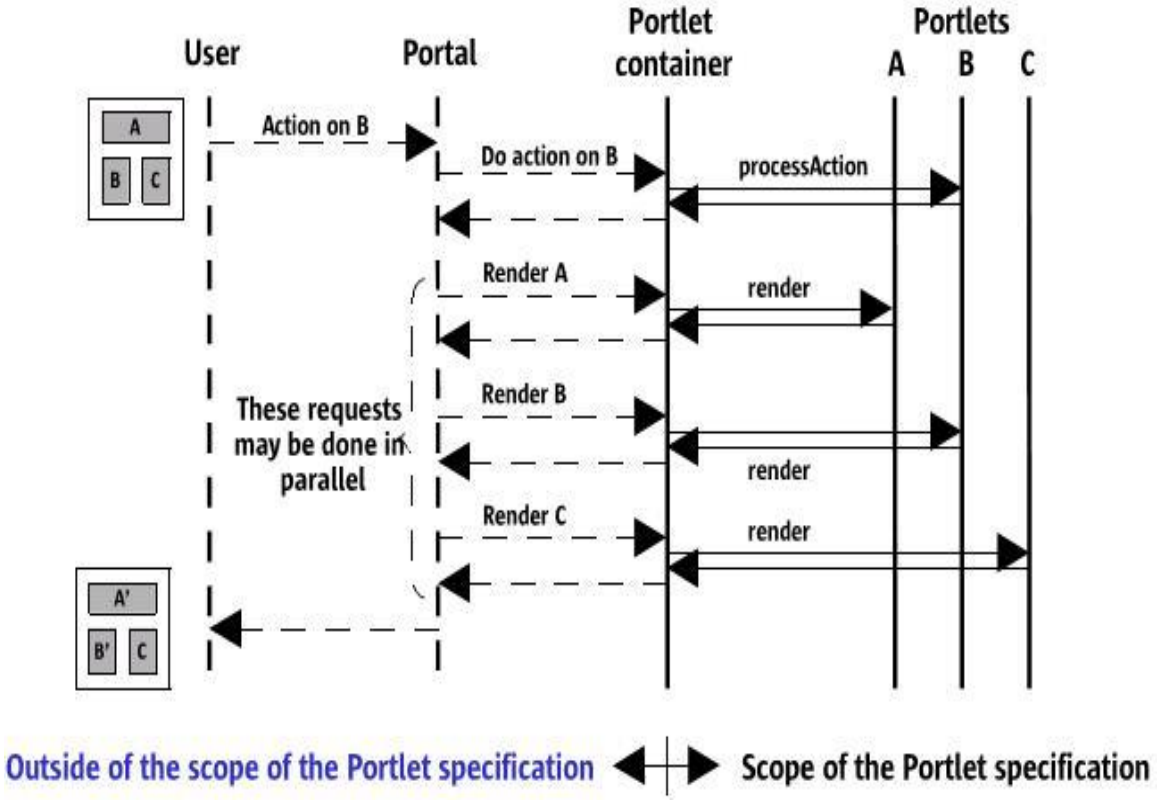
3. Quá trình tạo ra các trang portal

Các portlets chạy bên trong 1 portlet container. Portlet container nhận nội dung được sản sinh từ các portlets. Một cách điển hình, portlet container nắm giữ nội dung gửi đến portal. Portal server tạo trang portal với nội dung được sản sinh bởi các portlets và gửi nó đến thiết bị máy khách(client) khi nó được hiển thị đến người dùng.



4. Quá trình xử lý các yêu cầu trang portal

- Một người dùng thông qua ứng dụng client (ví dụ trình duyệt web HTML hay một điện thoại thông minh có thể lướt web) sau khi đã được chứng thực, đưa ra 1 yêu cầu HTTP đến portal.
- Yêu cầu đó được nhận bởi portal.
- Portal xác định nếu có 1 hành động(action) nhắm đến các portlets được tích hợp trong portal hay không.
- Nếu có 1 action nhắm đến 1 portlet, thì cổng điện tử yêu cầu portlet container triệu gọi portlet để xử lý hành động.
- Portlet container nhận nội dung được sản sinh từ các portlets.
- Portlet container nắm giữ nội dung gửi đến portal.
- Portal server tạo trang portal với nội dung được sản sinh bởi các portlets và gửi nó đến thiết bị máy khách(client) khi nó được hiển thị đến người dùng.



Portlet

Khái niệm chung về Portlet

Một server portal quản lý các yêu cầu của client. Và có một portlet container để quản lý việc chạy các Portlets. Bên trong portal là Portlet API - phục tùng mệnh lệnh của portlet container chúng quản lý trạng thái thực thi của portlet. Portlet container đánh giá những portlet đó thành các fragments, hoặc là tạo yêu cầu (request) của portlet hoặc là lấy một fragment trong cache. Sau đó, container nắm fragment gửi đến portal server để kết hợp chúng vào trong trang portal. Tạo nên giao diện của trang Portal.

Portlet Interface và lớp GenericPortlet

Giao diện portlet định nghĩa(cách thức) thái độ mà tất cả các portlet phải implement. Một cách cụ thể, bạn nên kế thừa (extends) lớp GenericPortlet để xây dựng portlet, bởi nó cung cấp kiến trúc chứa tất cả những phương thức cài đặt portlet điển hình.

Vòng đời của portlet

Rất giống như servlet, vòng đời một portlet được quản lý bởi container, và có phương thức init(khởi tạo) nó được dùng để quản lý những yêu cầu khởi tạo(tạo tài nguyên, cấu hình, vv...). Portlet chỉ được tải về khi cần đến, trừ khi bạn cấu hình container để tải chúng ngay khi khởi động. Phương thức init lấy một đối tượng object đã cài đặt(implement) lớp giao tiếp interface PortletConfig, cái quản lý các tham số khởi tạo và bó tài nguyên của Portlet: ResourceBundle. Đối tượng này có thể được sử dụng để lấy tham chiếu đến Object đã cài đặt(implement) lớp giao tiếp PortletContext interface. Nhà phát triển portlet không hoàn toàn mất thì giờ lo lắng về sự phức tạp của biệt lệ khởi tạo(exception) của portlet container, bởi thông thường chúng được ném ra, và nhà phát triển tác động trở lại lên chúng (gỡ rối những tình huống có thể dẫn đến biệt lệ exception và sửa chúng nếu có thể).

Trong khi khởi tạo, đối tượng portlet có thể ném ra 1 UnavailableException hay 1 PortletException. Trong trường hợp này, portlet không được kích hoạt mà nó bị giải phóng, việc khởi tạo không thành công. Portlet container có thể cố gắng instante và khởi tạo các portlets vài lần nữa sau khi thất bại.

Lỗi UnavailableException chỉ ra rằng portlet không sẵn sàng(unavaiable) trong một khoảng thời gian tối thiểu. Nếu gặp lỗi này, portlet container phải

chờ cho thời gian này qua đi mới instantiate và khởi tạo một portlet mới. Còn lỗi RuntimeException được ném ra trong quá trình khởi tạo có thể được xem như là một PortletException.

Phương thức destroy cung cấp để xoá hết các tài nguyên được thiết lập ở phương thức init(khởi tạo). Điều này tương tự với portlet destroy trong servlet, và được gọi mỗi khi container tống khứ portlet. Khi một exception được ném ra trong phương thức init của portlet, thì phương thức destroy được đảm bảo là không được gọi.

Tuy nhiên, nếu tài nguyên được tạo trong phương thức init() trước khi exception được ném, nhà phát triển không thể mong đợi phương thức destroy dọn dẹp chúng, mà phải quản lý chúng trong khối try-catch exception.

Các trạng thái thực thi portlet (Portlet Runtime States)

Khi một portlet đang chạy, nó có một đối tượng Preferences kết hợp cho phép tuỳ biến portlet. Những giá trị khởi tạo của Preferences được xác định trong mô tả triển khai(deployment descriptor), nhưng portlet có một sự truy cập đầy đủ một cách hệ thống đến tham chiếu của nó. Khi một portlet được đặt vào một trang, một Preferences sẽ tham chiếu đến nó. Sự kết đôi của portlet và đối tượng Preferences trên một trang được biết đến như là của sổ portlet.

Một trang có thể bao gồm rất nhiều những của sổ portlet như nhau bên trong hiển thị của nó. Trước khi bạn bắt đầu thắc mắc tại sao tất cả các đối tượng tham chiếu Preferences Object này là cần thiết, hãy hình dung rằng điều đó cung cấp khả năng để thao tác cho tính năng chủ yếu của portal-customization(khả năng tuỳ biến) .

Trong khi đối tượng tham chiếu khởi tạo portlet(Preferences Object) được tạo để xác định cấu hình và trạng thái thực thi của portlet, việc ngắt trạng thái để quản lý tuỳ biến giao diện của portlet là điều cần thiết. Chẳng hạn, nói bạn có một portlet thư mục làm công(employee directory portlet). Hiển nhiên là, nó cần một vài tham chiếu mới có thể chạy được. Tuy nhiên, khi employee directory portlet được nhúng vào trong trang chủ của "Bộ Tư pháp", nên không chỉ có một giao diện tuỳ biến, nhưng cũng có tham chiếu liên quan đến thực tế trên trang, chẳng hạn chỉ hiển thị các nhân viên của Bộ Tư pháp.

Quản lý yêu cầu portlet (Portlet Request Handling)

Có hai loại yêu cầu (request) có thể đưa ra đối với một portlet : action request và render request(yêu cầu hành động và yêu cầu hồi đáp). Không

ngẫu nhiên mà những yêu cầu(request) này đi cùng với các loại URL tương ứng: action URLs và render URLs. Một action URL nhắm tới phương thức processAction của portlet trong khi render URL hướng tới phương thức render của nó.

1. "Chỉ có thể là một"

Nếu yêu cầu của client là action request, thì nó chỉ hướng đến một portlet , cái sẽ phải thực thi trước tiên. Không có các action request khác có thể được thực thi trên portlet còn lại, chỉ có render request. Portlet container sẽ thực thi phương thức processAction() trên portlet đích, chờ đợi cho đến khi nó kết thúc trước khi nó thực thi hồi đáp (render) của những portlets còn lại trên trang. Việc gọi phương thức hồi đáp render trên các portlets còn lại có thể hoàn tất theo thứ tự, và có thể hoàn tất song song.

Phương thức processAction() chịu trách nhiệm việc thay đổi trạng thái trên một portlet cho trước, trong khi phương thức render chịu trách nhiệm sản sinh nội dung trình bày tương ứng(thích hợp) của portlet.

Vì thế, hoàn toàn hợp lý khi một user có thể thay đổi chỉ một portlet tại một thời điểm (bạn chỉ có thể click trên một hộp), và rằng tất cả các portlets phải gọi hồi đáp (render) để sản sinh lại nội dung của chúng trên kết quả của action. Tuy nhiên, đó không phải để nói rằng tất cả các portlet không thể thay đổi tại thời gian đã cho.

Hãy xem xét ví dụ chung sau: một portal cho Simpsons. Một trong những portlet cho phép bạn chọn các đặc tính của Simpson ở những trang mà bạn muốn xem. Những portlet khác chứa đựng những thông tin đặc tính, hình thức vừa rồi, những câu trích dẫn lớn nhất. Khi bạn chọn một đặc tính mới, bạn sẽ thay đổi trạng thái mà những đặc tính đã chọn hay portlet thông qua phương thức processAction() . Trong phương thức này, qua nó, bạn sẽ soạn thảo thuộc tính chia sẽ cho trước nó xác định đặc tính của trang mà bạn tác động, chúng sẽ là nguyên nhân tất cả các portlets tự hồi đáp cho những đặc tính trên khi bạn triệu gọi phương thức render của chúng.

Ghi nhớ một biệt lệ(exception) để khi một phương thức hồi đáp(render) của portlet được gọi, và khi nội dung của portlet bị giữ. Portlet API cho phép những containers chọn lựa để sử dụng bản copy nội dung được lưu giữ, thay vì gọi phương thức render. portlet container không là nơi cung cấp một cách dễ dàng cache, nhưng là nguồn đầu cơ (spec) cung cấp dễ dàng nơi lưu trữ kết thúc, mà được cấu hình trong mô tả triển khai ứng dụng portlet (deployment descriptor). Người triển khai cung cấp một yếu tố kết thúc-lưu trữ trong đó user xác định số giây lưu trữ (hoặc -1 nếu không kết thúc).

Nơi lưu trữ là 1client = 1 portlet, và không thể chia sẽ thông qua những yêu cầu của client. Tất nhiên là một nhà phát triển có thể implement portlet của anh ta do cache quản lý trong phương thức render, lưu trữ dữ liệu thường được yêu cầu trong PortletContext.

2. ActionRequest:

Như đã đề cập ở trên trong phần thảo luận về quản lý yêu cầu portlet, những yêu cầu hành động (action request) nắm giữ việc thay đổi trạng thái của 1 portlet dựa trên tham số yêu cầu hành động (action request). Nó được hoàn tất bằng cách sử dụng phương thức processAction(), nó lấy tham số là 2 đối tượng ActionRequest và ActionResponse . Đối tượng ActionRequest tương tự như đối tượng ServletRequest cho biết:

+ Các tham số yêu cầu hành động (action request)

+ Chế độ portlet

+ Phiên làm việc của portlet

+ Trạng thái cửa sổ

+ Các đối tượng tham chiếu portlet

+ Ngữ cảnh portal

Để thay đổi chế độ portlet hay trạng thái cửa sổ, bạn gọi phương thức tương ứng trong đối tượng ActionResponse . Sự thay đổi trở nên hiển nhiên khi phương thức render được gọi tiếp sau khi kết thúc quá trình xử lý trong phương thức processAction() . Bạn cũng có thể truyền các tham số render bằng cách sử dụng đối tượng ActionResponse.

3. RenderRequest:

RenderRequests sản sinh 1 fragment từ trạng thái hiện tại của portlet. Nó cung cấp:

+ Các tham số yêu cầu hồi đáp (render request)

+ Chế độ portlet

+ Phiên làm việc của portlet

+ Trạng thái cửa sổ

+ Các đối tượng tham chiếu portlet

Cũng có phương thức RenderResponse() đi kèm, nó cung cấp phương tiện cần thiết để hồi đáp nội dung. Bạn có thể gọi getOutputStream() hay getWriter() như từng làm trong servlet, hay bạn có thể gửi đi (dispatch) sự

sản sinh nội dung cho 1 servlet hay JSP. Ta sẽ đi chi tiết vào kỹ thuật này sau.

Các đối tượng request và response đều không là luồng an toàn. Điều đó có nghĩa là 1 nhà phát triển nên tránh chia sẽ các tham chiếu cho chúng với những luồng thực thi khác. Hầu hết các nhà phát triển sẽ không chú ý đến vấn đề này, nhưng hãy nhớ ghi chú nhỏ lý thú này lần sau khi bạn quyết định cố gắng làm điều gì đó không hợp lý.

Lớp GenericPortlet:

Lớp GenericPortlet là một lớp trừu tượng được cài đặt (implement)của giao diện portlet (interface). Đó là con đường chung nhất mà hầu hết users sẽ sử dụng để viết portlets - bằng cách kế thừa lớp này. Lớp GenericPortlet kế thừa phương thức render bằng cách cài đặt tiêu đề của portlet, và sau đó gọi phương thức doDispatch() của nó, và đến lượt nó, xác định chế độ của portlet, và gọi phương thức thích hợp : doEdit() để EDIT, doView() để VIEW. Ta sẽ thảo luận về chế độ portlet sau. Đoạn mã sau mô tả 1 lớp kế thừa GenericPortlet:

```
package org.opensourceportals.samples;
import java.io.IOException;
import javax.portlet.ActionRequest;
import javax.portlet.ActionResponse;
import javax.portlet.GenericPortlet;
import javax.portlet.PortletException;
import javax.portlet.PortletMode;
import javax.portlet.PortletRequestDispatcher;
import javax.portlet.RenderRequest;
import javax.portlet.RenderResponse;

/**
* ExamplePortlet là ví dụ cơ bản về 1 lớp kế thừa GenericPortlet
*/

public class ExamplePortlet extends GenericPortlet {
/*
* phương thức này sẽ ghi đè phương thức doEdit của GenericPortlet
*Được gọi để cung cấp 1 đánh dấu được hồi đáp khi chế độ portlet là
PortletMode.EDIT
```



```
* Lúc này, ta sẽ gửi ph/thức đến 1 JSP trong thư mục gốc của portlet gọi là
"edit.jsp"
*/
protected void doEdit(RenderRequest request,RenderResponse response)
throws PortletException, IOException {
PortletRequestDispatcher prd
=getPortletContext().getRequestDispatcher("/edit.jsp");
prd.include(request, response);
}

Ta sẽ mô tả ExamplePortlet, có kế thừa lớp GenericPortlet. Ở đây ta có thể
ghi chồng phương thức doEdit, nó quản lý việc hồi đáp khi portlet ở chế độ
EDIT.

/*
* phương thức này sẽ ghi đè phương thức doEdit của GenericPortlet
*Được gọi để ccấp 1 đánh dấu được hồi đáp khi chế độ portlet là
PortletMode.HELP
* Lúc này, ta sẽ gửi ph/thức đến 1 JSP trong thư mục gốc của portlet gọi là
"help.jsp"
*/
protected void doHelp(RenderRequest request,RenderResponse response)
throws PortletException, IOException {
PortletRequestDispatcher prd =
getPortletContext().getRequestDispatcher("/help.jsp");
prd.include(request, response);
}

/*
* Phương thức này sẽ ghi đè phương thức doEdit của GenericPortlet
*Được gọi để cung cấp1 đánh dấu được hồi đáp khi chế độ portlet là
PortletMode.VIEW
* Lúc này, ta sẽ gửi ph/thức đến 1 JSP trong thư mục gốc của portlet gọi là
"view.jsp"
*/
protected void doView(RenderRequest request,RenderResponse response)
throws PortletException, IOException {
PortletRequestDispatcher prd
=getPortletContext().getRequestDispatcher("/view.jsp");
```



```
prd.include(request, response);}
```

Tương tự, ta cung cấp các cách ứng xử được yêu cầu để hồi đáp portlet khi nó ở chế đội HELP và VIEW.

```
/* phương thức này được ghi chồng để xác định tiêu đề một cách tự động.
Nó có thể có
* ích nếu bạn đang có tham số trong tiêu đề của bạn giống như : "News on
16/11/2005"
*/

protected String getTitle(RenderRequest request) {
return "Example Portlet";
}

/* Đây là phương thức cốt yếu của portlet , các thao tác của trạng thái
portlet
* được hoàn tất thông qua phương thức này. Để đơn giản, chúng ta sẽ
truyền
* đối số chỉ định chế độ portlet mà portlet có thể được thiết lập.
*/

public void processAction(ActionRequest request,ActionResponse response)
throws PortletException, IOException {
PortletMode mode =new PortletMode(request.getParameter("mode"));
response.setPortletMode(mode);
}
}
```

Cuối cùng, ta chỉ định ghi chồng phương thức getTitle(), cho phép hoàn toàn hợp lý trong việc hồi đáp tiêu đề (như hiển thị ngày hiện tại) thay vì hiển thị tiêu đề tĩnh được mô tả trong mô tả triển khai (deployment descriptor). Ta cũng có thể quản lý phương thức processAction(), nó chịu trách nhiệm trong cách trả lời đối tượng ActionRequest.
Đoạn mã nêu trên chỉ ra sự cài đặt cơ sở của portlet bằng cách viết 1 lớp kế thừa lớp GenericPortlet. Portlet này không gửi đi đến các trang JSPs khác dựa trên chế độ của nó (và thiết lập tên của nó một cách hệ thống), nhưng bạn gặp mối nan giải của việc cài đặt portlett.

Các yếu tố khác của Java Portlet API

Giờ thì bạn đã khảo sát những khái nhiệm ở mức cao của portlet API, đoạn này sẽ cho biết các thành phần ở mức thấp bên trong đặc tả, việc cung cấp 1 portlet hình cảnh của nhà phát triển ở bên trong của đặc tả, làm sáng tỏ những khái niệm quan trọng và những nguy cơ tiềm ẩn.

PortletConfig

Khi 1 portlet được khởi tạo, nó cần truy cập đến những tham số khởi tạo và thông tin cấu hình khác. Đối tượng PortletConfig cung cấp những thứ đó. Thêm vào những thông số khởi tạo, đối tượng PortletConfig còn có thể trình bày 1 ResourceBundle (bó tài nguyên) cho portlet.

ResourceBundle bao gồm 1 vài trường được yêu cầu bởi đặc tả, bao gồm tiêu đề, tiêu đề ngắn, và các từ khoá.

1 ResourceBundle cho phép việc định vị ứng dụng portlet dễ dàng hơn. Bạn có thể xác định ResourceBundle trong dòng của mô tả triển khai ứng dụng portlet (deployment descriptor) như sau:

```
<portlet>
...
<portlet-info>
<title>Homer’s D’oh a Day Portlet</title>
<short-title>doh</short-title>
<keywords>Simpsons, Homer Simpson, Entertainment</keywords>
</portlet-info>
...
</portlet>
```

Như 1 sự lựa chọn, bạn có thể chỉ định 1 tham chiếu đến 1 ResourceBundle theo cách:

```
<portlet>
...
<portlet-info>
<resource-bundle>com.somedomainname.HomerPortlet</resource-
bundle>
</portlet-info>
...
</portlet>
```


Bất cứ cách nào bạn sử dụng (cái đầu tiên thường tốt hơn cho các ứng dụng với yêu cầu định vị tối thiểu), các hiệu ứng mạng nhện cho người phát triển cũng như vậy. Những tính chất này thường được tạo bên trong một ResourceBundle và được làm hiệu lực thông qua đối tượng PortletConfig.

PortletURL

Khi xây dựng nội dung của portlet, điều cần thiết là xây dựng các URLs cung cấp khả năng gọi portlet. Đây là cái cơ bản tạo nên tương tác trong portal. Nhằm mục đích cho phép việc tạo lập PortletURL đúng thực tế, có 2 sự cài đặt: ActionURL và RenderURL. Cả 2 đều được tạo từ giao diện RequestResponse interface bằng cách sử dụng các phương thức createActionURL() và createResponseURL() tương ứng.

ActionURL: cung cấp khả năng đưa ra những yêu cầu hành động (action request) đến portlet đích trên portal, để làm những việc như thay đổi chế độ portlet, thay đổi trạng thái cửa sổ, xác thực 1 form, v.v...

RenderURL: cung cấp khả năng bỏ qua phương thức processAction() của portlet và đơn thuần triệu gọi phương thức hồi đáp (render), việc truyền thông số hồi đáp để điều khiển việc biểu diễn.

Những nhà phát triển portlet nên sử dụng đối tượng PortletURL (hoặc các thư viện thẻ đi kèm) thay vì thao tác trực tiếp trên những dòng truy vấn HTTP. Điều tất yếu là nhà phát triển không nên dùng GET trong các form HTML. Đó là bởi vì portal có thể mã hoá các thông số trạng thái nội tại bên trong PortletURL.

Bạn có thể tìm thấy rất nhiều phương thức trong lớp giao tiếp PortletURL interface đáng chú ý:

- ***setSecure()***: cung cấp khả năng để chỉ định URL ở đâu nên dùng HTTP ở đâu thì không. Nếu nó không được sử dụng, nó tiếp tục với bất cứ thứ gì yêu cầu hiện tại chỉ định. Vì thế, bạn không phải chỉ định lặp lại nó.
- ***setWindowState()***: cho phép bạn thay đổi trạng thái cửa sổ của portlet.
- ***addParameter()***: thêm các thông số vào URL.
- ***toString()***: cung cấp 1 chuỗi biểu diễn của URL. Chú ý rằng nó không đảm bảo 1 URL hợp lệ, như là portal có thể sử dụng token để viết lại URL.
- ***setPortletMode()***: cho phép bạn thiết lập chế độ của portlet .

Các chế độ của portlet

1 chế độ portlet biểu diễn 1 trạng thái chức năng của 1 portlet. Đây được dùng bởi portlet để xác định làm thế nào để quản lý các yêu cầu hồi đáp. Điều đó, phụ thuộc vào chế độ, portlet sẽ hồi đáp những đánh dấu khác nhau.

Các portlets có thể thay đổi chế độ của chúng như là 1 phần của quá trình xử lý yêu cầu hành động (action request). Thêm vào đó, 1 portlet có thể được cấu hình với những chế độ thích hợp khác nhau và hạn chế xa hơn tính sẵn sàng dựa trên chức năng. Bảng sau mô tả các chế độ portlet chuẩn được định nghĩa trong portlet API.

Các chế độ của Portlet:

- ***VIEW*** Sản sinh đánh dấu hình dung trạng thái và tính chất của portlet. Nhà phát triển cài đặt doView() của lớp GenericPortlet để cung cấp chức năng này.
- ***EDIT*** Sản xuất đánh dấu để có thể thay đổi tính chất của portlet. Nhà phát triển cài đặt doEdit() của lớp GenericPortlet để cung cấp chức năng này.
- ***HELP*** Cung cấp các dòng hướng dẫn cho portlet. Nhà phát triển cài đặt doHelp() của lớp GenericPortlet để cung cấp chức năng này.

Một portal cũng có thể cung cấp các chế độ portlet tuỳ thích. Nhớ rằng điều đó phụ thuộc portal, không phụ thuộc portlet. Vì thế, nếu 1 portlet cài đặt chế độ portlet bổ sung, thì chúng sẽ không chuyển được giữa các container portlet khác nhau. portlet cần phải ghi đè phương thức doDispatch() để gọi phương thức hồi đáp thích hợp.

Chẳng hạn, nếu bạn định nghiã 1 chế độ portlet có tên "SPECIAL" phương thức doDispatch() sẽ gọi phương thức hồi đáp của nó, thông thường là doSpecial(). Tất nhiên là, vì bạn đang cài đặt phương thức, bạn có thể gọi bất cứ phương thức gì bạn muốn.

Cũng ghi chú rằng bạn có thể chỉ định với loại đánh dấu nào là thích hợp cho 1 chế độ portlet cho trước.

Các cửa sổ trạng thái

Các cửa sổ trạng thái chỉ định portlet bao nhiêu không gian được sử dụng cho nó. Điều này cho phép portlet chỉnh sửa những hồi đáp của nó để thích

hợp với trạng thái của cửa sổ. Bảng sau bao gồm những trạng thái cửa sổ được xác định bằng Portlet API.

Trạng thái

- ***NORMAL*** portlet sẽ chia màn hình với các portlet khác. Điều đó có nghĩa là portlet sẽ giới hạn các đánh dấu của nó (markup).
- ***MINIMIZED*** portlet sẽ cung cấp ít đầu ra hoặc không.
- ***MAXIMIZED*** portlet không chia sẻ màn hình với các portlet khác, vì thế portlet không bị giới hạn trong đánh dấu của nó (markup).

Cũng giống như chế độ portlet, 1 portlet có thể định nghĩa các cửa sổ trạng thái tuỳ ý, nó cũng phải được cấu hình trong mô tả triển khai portlet (portlet deployment descriptor).

Ngữ cảnh portlet

PortletContext là 1 đối tượng wrapper cho ứng dụng portlet của bạn. Có 1 đối tượng PortletContext cho mỗi ứng dụng portlet. PortletContext cung cấp:

- Truy xuất biến khởi tạo
- Lấy và thiết lập các thuộc tính ngữ cảnh
- Ghi lại sự kiện
- Lấy tài nguyên ứng dụng (như hình ảnh, file XML ...)
- Lấy được 1 yêu cầu gửi đi để có sức mạnh đòn bẩy của servlet và JSPs trong portlet.

Ngữ cảnh portal

Một portlet có thể lấy 1 tham chiếu đến portal mà nó đang chạy trong đó thông qua đối tượng PortalContext. Việc gọi phương thức getPortalContext() của đối tượng PortletRequest sẽ trả về PortalContext .

PortalContext cung cấp :

- Tên của portal - thông qua phương thức getPortalInfo()
- Các đặc tính của portal - qua phương thức getProperty() và getPropertyNames()
- Những chế độ portlet được hỗ trợ - qua getSupportedPortletModes()

- Những cửa sổ trạng thái được hỗ trợ - qua getSupportedWindowStates()

Các tham chiếu portlet (Portlet Preferences)

Nhằm thao tác tuỳ biến hoá và cá nhân hoá portlet của bạn, bạn cần thay đổi chút ít các tham số của portlet bằng nhiều cách. Những cái này gọi là portlet preferences (tham chiếu của portlet). Ví dụ cái portlet cổ điển là cái portlet thời tiết, và theo sau ví dụ đó, bạn có thể có 1 tham chiếu có tên "cities", với các giá trị biểu diễn zip code của những thành phố mà bạn cần biết thời tiết. Ghi chú rằng các tham chiếu chỉ dành cho việc cấu hình portlet, vì thế việc bảo trì danh sách tất cả các thành phố thích hợp trong 1 tham chiếu sẽ là không thích hợp cho dùng các preferences. Việc suy nghĩ về chúng theo ý nghĩa của lập trình hướng đối tượng, chúng sẽ là các thuộc tính của bản thân các portlets.

Preferences được thao tác thông qua 1 đối tượng Object có cài đặt (implement) lớp giao tiếp PortletPreferences interface. Tất cả các giá trị tham chiếu được lưu trữ trong mảng String, vì thế bạn sẽ không thể lưu trữ toàn Object như bạn làm với thuộc tính yêu cầu hay phiên làm việc, cũng không có những thuận lợi trong khai báo kiểu, như bạn làm với các lối vào môi trường. Vì thế, nếu bạn đang lưu trữ các số dưới các tham chiếu, bạn sẽ cần tự chuyển đổi.

Đặc biệt, lớp giao diện PortletPreferences interface cung cấp:

- ***getNames():*** sẽ trả về 1 bảng liệt kê Enumeration các tên các preferences thích hợp.
- ***getValue():*** bạn truyền cho nó tên của tham chiếu bạn quan tâm, đi kèm với 1 giá trị mặc định (trong trường hợp nó không tìm thấy), và nó trả về phần tử đầu riên của mảng các giá trị tham chiếu.
- ***getValues():*** bạn truyền cho nó tên tham chiếu bạn muốn và 1 mảng String các giá trị mặc định và nó trả về 1 mảng String của các giá trị của nó.
- ***setValue():*** bạn truyền cho nó tên tham chiếu và giá trị của tham chiếu đó, và nó thiết lập (cài đặt) nó là 1 phần tử đơn mảng String chứa đựng các giá trị. phương thức này ném ra 1 biệt lệ nếu tham chiếu không bị thay đổi UnmodifiableException .

- ***setValues():*** bạn truyền cho nó tên tham chiếu và 1 mảng String biểu diễn giá trị của tên đó, và nó thiết lập các giá trị tham chiếu. phương thức này ném ra 1 biệt lệ nếu tham chiếu không bị thay đổi UnmodifiableException .
- ***isReadOnly():*** bạn truyền cho nó tên tham chiếu và nó trả về 1 giá trị Boolean xác định tahm chiếu có thể thay đổi được hay không.
- ***reset():*** bạn truyền cho nó tên tham chiếu và nó trả về 1 giá trị mặc định, nếu không, nó sẽ xoá tham chiếu.
- ***store():*** lưu trữ các tham chiếu. Bởi đó chỉ có thể hoàn thành bên trong processAction(), nó sẽ ném ra 1 biệt lệ IllegalStateException nếu nó hoàn tất trong 1 lời triệu gọi hồi đáp, phương thức cũng sẽ ném ra biệt lệ ValidatorException nếu không hợp lệ.
- ***getMap():*** trả về map của các tham chiếu. Map bao gồm các khoá String và 1 mảng String[ ] chứa các giá trị. Map này cũng không thể thay đổi.

Ghi chú 2 điều quan trọng để hiểu về phương thức store() .

Trước tiên, đó là 1 giao dịch nguyên tử. Vì thế, tất cả các thay đổi phải thành công hoặc không thay đổi nào thành công. Đó là điểm then chốt để hiểu nếu bạn có 1 danh sách tham chiếu khổng lồ cho portlet của bạn và bạn không nhập 1 số lượng lớn khủng khiếp các đầu vào.

Hai là, bạn có thể lấy nọc cùng lúc ghi vào bộ lưu trữ tham chiếu. 1 thông báo critical ở đây là bạn nên xem những tham chiếu của bạn như 1 thực thể riêng biệt và không là 1 tập hợp các tham số độc lập có thể sử dụng được thông qua 1 giao diện chung.

Nhà phát triển có thể định nghĩa các lớp tuỳ ý để cung cấp 1 sự hợp lệ cho các tham chiếu. Những lớp này cài đặt(implement) lớp giao tiếp PreferencesValidator interface và phải cung các các khả năng hợp lệ trong nghĩa luồng - an toàn. Việc implement giao tiếp PreferencesValidator interface là rất đơn giản: chỉ chó 1 phương thức validate(), nó lấy đối tượng

PortletPreferences và không trả về gì cả. Đơn giản ném ra 1 biệt lệ ValidatorException nếu 1 vài giá trị không logíc với cái bạn định nghĩa.

Sessions (Phiên làm việc)

Bởi vì portal được xây dựng trên cơ chế request - response (và chủ yếu dựa trên giao thức HTTP), phải có 1 vài cơ chế thích hợp để bảo trì trạng thái qua các triệu gọi. Chẳng hạn, không thể thấy việc xác thực người dùng (authenticate users) với mọi yêu cầu. Có rất nhiều kĩ thuật để quản lý phiên làm việc sessions, với cookies và mã hoá URL là 2 cách thông dụng nhất của ứng dụng Web (cookies được dùng dưới các hood bởi nhiều servlet container để implement HTTPSession interface).

Session là điểm then chốt đối với nhà phát triển portlet, nhưng có nhiều các để cài đặt chúng (implement), vì thế portlet API cung cấp giao tiếp PortletSession interface. Khi 1 client tạo 1 request, server gửi đi 1 định danh session trong response. Nếu client muốn kết nối vào session (phiên làm việc), client cung cấp định danh session đó cho các request tiếp sau của họ.

PortletSession có thể được sử dụng để nắm giữ các thuộc tính. PortletSession hoạt động giống như HTTPSession trong mặt này, việc cung cấp khả năng lưu trữ cặp đôi khoá - giá trị (key-value), với đối tượng bất kỳ. Có 1 điểm khác nhau chủ yếu:

PortletSession có 2 phần khác nhau:

- APPLICATION_SCOPE thì tương tự như phạm vi HTTPSession. 1 đối tượng được đặt rong phạm vi này là thích hợp cho tất cả các portlets bên trong phiên làm việc (session).
- PORTLET_SCOPE chỉ đến khi 1 đối tượng là thích hợp chỉ 1 portlet đó. PORTLET_SCOPE là thống nhất trong cách đặt không gian tên cho 1 thuộc tính cho trước. Ví dụ, 1 thuộc tính nó tên là city.name sẽ được lưu trữ trong javax.portlet.p.47?city.name. ("47" là số định dạng được gán bên trong). Tên thuộc tính này ngăn cản namespace (không gian tên) xung đột với các portlets khác lưu trong biến session của chúng với các tên tương tự.

Mặc dù việc có 1 hệ thống đặc biệt để đặt tên những thuộc tính của nó, PORTLET_SCOPE lại không bảo vệ các thuộc tính của nó với các thành phần web khác. Thêm vào đó, ứng dụng namespace (không gian tên) là hoàn toàn thực hiện dưới hood. Bạn chỉ việc gọi các phương thức getAttribute()

và setAttribute() để chỉ định PORTLET_SCOPE và sự chuyển đổi không gian tên namespace được thực hiện bởi đối tượng PortletSession. Để làm cho nó thuận lợi hơn, các thành phần web khác có thể nhận tính năng này thông qua phương thức PortletSessionUtil.decodeAttribute() bằng cách truyền tên thuộc tính đơn giản, ví dụ như "city.name".

Gọi JSPs và servlet

Bởi các ứng dụng portlets hoàn toàn mỡ rộng của các ứng dụng web, ở đây phải có 1 cơ chế để include JSP và servlet trong portlet. Dựa vào việc nghiên cứu đầu tiên về lớp GenericPortlet, nhiều nhà phát triển Web nghĩ, " Ôi, không! chúng ta đã trở lại những ngày của servlet xa xưa với việc nhúng các đánh dấu (markup)". Tuy nhiên, cũng giống như servlet, nhà phát triển nhận thấy việc sử dụng phương thức RequestDispatcher() là có ích để tránh cái gọi là "đặt tất cả trứng của mình trong 1 cái rổ " tức vấn đề đặt tất cả các markup của mình trong 1 servlet, và tất cả mã Java trong trang JSP, 1 nhà phát triển portlet có thể dùng PortletRequestDispatcher.

Khi cài đặt (implement) phương thức hồi đáp của giao diện portlet hay đúng hơn là cài đặt 1 trong các phương thức "do__" của lớp GenericPortlet (chẳng hạn như doView, doEdit, doHelp ...), nhà phát triển có thể dùng PortletRequestDispatcher như sau:

```
String reqPath = “/calView.jsp”;
PortletRequestDispatcher prd =
portContext.getRequestDispatcher(reqPath);
prd.include(req, resp);
```

Trong trường hợp này, ta phải chỉ định trang JSP của mình: calView.jsp, nó sẽ nằm trong thư mục gốc của ứng dụng portlet, mà ta sẽ chỉ tới với các dấu slash /. Bạn phải luôn bắt đầu đường dẫn với dấu slash hở /, và cung cấp 1 đường dẫn từ thư mục gốc của PortletContext (thường là thư mục gốc của ứng dụng portlet). Từ đó, bạn lấy PortletRequestDispatcher (prd) từ phương thức PortletContext (portContext). Sau đó bạn truyền cho các phương thức RenderRequest(req) và RenderResponse(resp) các tham số vào phương thức include của PortletRequestDispatcher(prd).

Tương tự, ta có thể gọi 1 servlet bằng cách mô tả tên của nó (trong file mô tả triển khai ứng dụng Web web.xml). Chẳng hạn, ta có thể phải chỉ định 1 servlet như sau trong web.xml:


```
<servlet>
<servlet-name>chart</servlet-name>
<servlet-class>org.opensourceportals.ChartServlet</servlet-class>
<load-on-startup>0</load-on-startup>
</servlet>
```

Bởi vì ta đã đặt tên servlet của mình là "chart", nên ta có thể chỉ định nó như sau :

```
String reqName = “chart”;
PortletRequestDispatcher prd =
portContext.getNamedDispatcher(reqName);
prd.include(req, resp);
```

Lúc này ta dùng phương thức getNamedDispatcher () với tên của servlet để lấy 1 Object PortletRequestDispatcher. Đây là 1 điểm quan trọng khác để ta xem xét nếu chọn phải làm như sau:

```
String reqPath = “/calView.jsp?user=Jennifer”;
PortletRequestDispatcher prd =
portContext.getRequestDispatcher(reqPath);
prd.include(req, resp);
```

Vì tham số user được chỉ định trong chuỗi truy vấn, nó sẽ lấy theo thứ tự từ trước đến sau vài tham số hồi đáp khác được đặt tên user và truyền cho JSP. Bạn hầu như chắc chắn sẽ không chú ý đến vấn đề này, nhưng đó là những thứ phải ghi nhớ trong trường hợp bạn rơi vào cách ứng xử : việc chỉ định một chuỗi truy vấn để lấy lần lượt các tham số khác. Có nhiều hạn chế trong việc dùng Object HttpServletRequest. Những phương thức này không thích hợp để include servlet và JSP:

- getProtocol()
- getRemoteAddr()
- getRemoteHost()
- getRealPath()

- getRequestURL()
- getCharacterEncoding()
- setCharacterEncoding()
- getContentType()
- getInputStream()
- getReader()
- getContentLength()

Những phương thức sau đều trả về null (trừ getContentLength trả về 0) nếu bị triệu gọi từ 1 servlet hay JSP được include bởi 1 Object PortletRequestDispatcher. Tuỳ thuộc vào việc làm thế nào bạn tạo PortletRequestDispatcher, bạn có thể không phải truy xuất đến các phương thức khác. Nhưng ph/th ức bổ sung này là không thích hợp việc truy xuất servlet và JSPs từ PortletRequestDispatcher được tạo bằng cách gọi getNamedDispatcher().

- getPathInfo()
- getPathTranslated()
- getServletPath()
- getRequestURI()
- getQueryString()

Lý do tại sao những ph/thức trên sẽ không thích hợp khi gọi phương thức getNamedDispatcher() khá đơn giản : Bởi bạn đã không sử dụng 1 đường dẫn cho những yêu cầu (request) của mình, không có dữ liệu mà các trường này có thể chứa. Tương tự như thế, HttpServletResponse cũng có những hạn chế trên cái gì có thể sử dụng được.

Những phương thức không sẵn có của HttpServletResponse:

- encodeRedirectURL()
- encodeRedirectUrl()
- setContentType()
- setContentLength()

- setLocale()
- sendRedirect()
- sendError()
- addCookie()
- setDateHeader()
- addDateHeader()
- setHeader()
- addHeader()
- setIntHeader()
- addIntHeader()
- setStatus()
- containsHeader()

Những phương thức mã hoá luôn trả về null, và containsHeader() luôn trả về trị false, nhưng những cái còn lại thì sẽ đơn giản không làm gì cả.

Java Portlet Specification đề nghị đối với việc sử dụng các phương thức tiếp theo của RequestDispatcher của các servlet và JSPs đã include. Trong khi điều này không có vẻ như 1 sự thoả thuận lớn, ghi chú là Apache's Struts Framework dùng phương thức forward RequestDispatcher trong Object ActionServlet của nó. Việc đưa ra thông dụng của Struts như là 1 framework ứng dụng Web, đó có thể làm nên 1 sự thúc đẩy (tác động) quan trọng trong việc thống nhất tích hợp portal trong nhiều cơ quan tổ chức. Điều đó không có nghĩa là nó không làm việc vô tổ chức, nhưng nó không là 1 định mệnh và sẽ được xem xét cẩn thận trong điều kiện của bạn và được testing với cài đặt portal của bạn.

Cấu tạo ứng dụng portlet

Ứng dụng portlet được cấu tạo giống như ứng dụng Web trong đó ta có các tính năng sau :

- Có thể bao gồm servlets, JSPs, classes, files JAR (java archives), và các file tĩnh khác.

- Được đóng gói - tất cả các ứng dụng Web đều được đóng gói trong thư mục root chung.
- Có thư mục WEB-INF/classes để lưu giữ các lớp độc lập có thể tải bởi classloader.
- Có thư mục WEB-INF/lib để lưu giữ Java Archives (JAR) có thể tải bởi classloader.
- Được đóng gói thành file Web Archive (WAR)

Thêm vào các tính năng này, ứng dụng portlet còn bao gồm 1 file mô tả triển khai ứng dụng Web, được đặt tại WEB-INF/portlet.xml.

Bảo mật

Bởi bảo mật là 1 vấn đề lớn hơn những yêu cầu của portlet API ta sẽ đề cập đến vấn đề này phần sau.

Các định nghĩa kiểu CSS (Cascading Stylesheets)

Nhằm đặt đến 1 giao diện chung và có thể plug cho các portlets, Java Portlet API định nghĩa tập các kiểu CSS (Cascading Stylesheets) mà portlet có thể sử dụng để hồi đáp các markup của chúng. Bằng cách sử dụng 1 tập các kiểu chuẩn, portal có thể hổ trợ skin, tuỳ biến hoá màu sắc và font chữ. Những kiểu nầy có nghĩa đồng thời với chuẩn OASIS Web Services for Remote Portlets (WSRP).

Kiến trúc của Portlet

1 portal bao gồm những portlet cá nhân người dùng, chúng có thể được thêm vào bớt ra và tuỳ biến bởi người dùng. Các portlet là cái làm đẹp cho tầng trình bày của portal. Chúng là thành phần giao diện người dùng có thể tuỳ biến bởi user, và chúng có thể được thêm vào và bớt đi từ portal 1 cách dễ dàng. Chúng cũng xử lý yêu cầu người dùng và sản sinh nội dung theo yêu cầu thông qua portlet container.

Portlet Container

Một portlet container được sử dụng để quản lý các portlet thông qua vòng đời của chúng. Container cho phép nhà phát triển gọi các phương thức chỉ định trong suốt thời gian sống của 1 portlet . Nó nâng các nhà phát triển lên quyết định phương thức nào được cài đặt (implement). Là 1 qui tắc tổng quát, các nhà phát triển nên thường kế thừa lớp GenericPortlet khi tạo các

portlet. Lớp GenericPortlet gọi các phương thức hồi đáp chỉ định dựa trên chế độ hiện tại (current mode) của portlet. Những phương thức này được mô tả trong bảng sau:

Phương thức

doEdit được gọi bởi phương thức hồi đáp khi portlet ở chế độ EDIT. Chế độ EDIT nên được dùng với mục đích chỉ định biên soạn portlet . Ví dụ : nếu ta 1 portlet danh mục vốn đầu tư chứng khoán và nó chứa đựng 1 danh sách các chứng khoán, khi ta nhấn edit ta sẽ có thể biên soạn danh sách này trong danh mục vốn đầu tư của mình (portfolio).

doView được gọi bởi phương thức hồi đáp khi portlet ở chế độ VIEW. Chế độ VIEW là chế độ chính cuả portlet. Nội dung chính của portlet sẽ hiển thị trong suốt chế độ này. Ví dụ : Nếu ta có 1 portlet xếp thứ hạng bóng đá, thứ hạng sẽ hiển thị trong chế độ VIEW.

doHelp được gọi bởi phương thức hồi đáp khi portlet ở chế độ HELP. Chế độ HELP dùng hiển thị các hướng dẫn chỉ định về cách sử dụng portlet. Lấy ví dụ danh mục vốn đầu tư chứng khoán, chế độ HELP có thể mô tả cáchs thích hợp để biên soạn (edit) và nhập các chứng khoán cũng như các kí hiệu của chúng vào trong portlet.

Các phương thức khác có thể được truy cập với hoặc không kế thừa từ lớp GenericPortlet. Những phương thức này cho phép nhà phát triển nắm giữ những chức năng khác nhau mà không cần thiết trên chế độ hiện tại của portlet . Những phương thức bổ sung này được trình bày trong bảng sau:

init() : được gọi bởi container khi portlet được tạo và được sử dụng để khởi tạo portlet và chuẩn bị để sử dụng nó. Lấy ví dụ, nếu portlet của bạn cần phải load những cấu hình chỉ định từ CSDL hay nguồn dữ liệu bên ngoài, thì phải mất nhiều thời gian thực hiện việc đó.

destroy() : được gọi bởi container khi container phá huỷ portlet , bằng cách cho phép bạn xoá sạch bất cứ thứ gì cần chú ý đặc biệt. Lấy ví dụ, nếu 1 kết nối CSDL được mở, thì nó cho phép bạn đóng những kết nối mở và xoá gọn gàng khi portlet bị phá huỷ.

processAction() : được gọi bởi container khi user xác nhận thay đổi trên portlet. Đây là phương thức thiết yếu để bạn có thể xử lý dữ liệu đã xác nhận bởi user từ 1 portlet. Ví dụ, bạn có thể có 1 form yêu cầu ngày sinh của user. Khi user xác nhận form, processAction() được gọi, cho phép bạn xử lý thông tin và quyết định hiển thị cái gì đến user, như việc thay đổi chế độ portlet (mode) nếu bạn muốn.

render() : được gọi bất cứ khi nào portlet cần vẽ lại. Trong hầu hết các trường hợp, bạn sẽ không cần nắm giữ phương thức này vì doView, doEdit, và doHelp tồn tại trong lớp GenericPortlet và tự động được gọi bởi phương thức render.

Bốn phương thức mà bạn sẽ tương tác nhiều nhất khi phát triển 1 portlet là doView(), doEdit(), doHelp(), và processAction() .Những phương thức này xuất hiện tương tự như servlet vì chúng gửi yêu cầu (request) và trả lời (response) các đối tượng. Tuỳ thuộc vào những phương thức được sử dụng với chúng, những Object này có thể được sử dụng để:

- Đạt được sự truy xuất đến đối tượng tham chiếu của portlet nhằm duy trì trạng thái.
- Lấy tham số được xác nhận bởi user thông qua form.
- Đạt được thông tin phiên làm việc (session).
- Thu thập thông tin bảo mật về user, như thông tin user-role.
- Thay đổi những biểu hiện khác nhau của portlet, và làm thế nào nó được hồi đáp. Chẳng hạn, tiêu đề có thể bị thay đổi thông qua đối tượng response.

