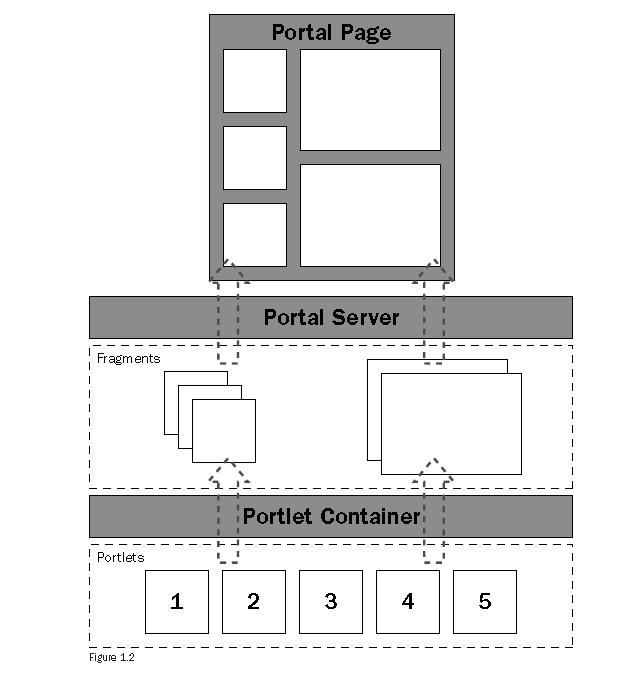
**CÁC KHÁI NIỆM**

**Portlets** là các thành phần hiển thị thông tin liên quan cho người dùng. Các ứng dụng cho portlets bao gồm việc hiển thị Thư điện tử, các báo cáo thời tiết, các diễn đàn thảo luận, và tin tức…

Các chuẩn Portlet nhằm cho phép các nhà phát triển phần mềm tạo ra các portlets mà có thể "cắm" vào bất cứ Portal nào có hỗ trợ các chuẩn đó.

Mục đích của các interface của Dịch vụ Web cho Portlets từ xa là cung cấp một chuẩn dịch vụ web mà cho phép khả năng "plug-and-play" của các Portal, hoặc làm trung gian cho các ứng dụng web mà kết hợp nội dung, và các ứng dụng từ nhiều nguồn khác nhau.



Bản miêu tả chi tiết kĩ thuật của cổng điện tử cho phép tương tác qua lại giữa các portlets và Portal.

Điểm giống nhau portlet và serverlet:

- Đều là những web components chạy trên nền J2EE

- Đều được quản lý bởi các container, quản lý các giao tác

- Họat động theo mô hình request – response

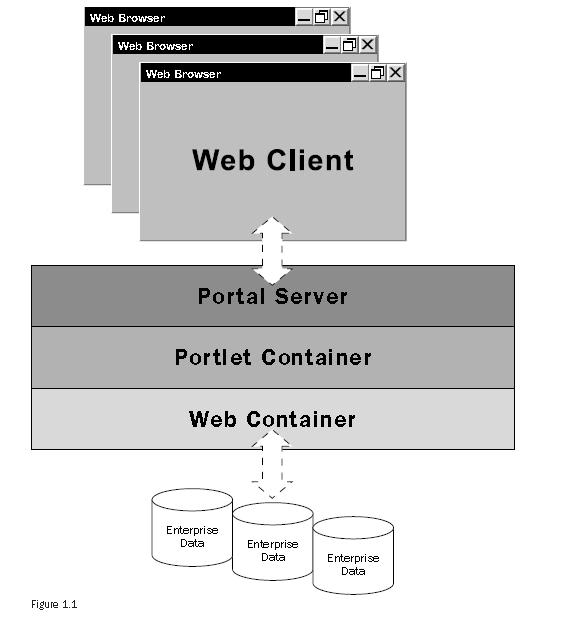
Điểm khác biệt:

- Portlet sinh ra các fragment (một phân đoạn của trang hiển thị), trong khi servlet sinh ra toàn bộ trang hiển thị

- Không như servlet có thể được gọi trực tiếp thông qua url, portlet không được gọi trực tiếp bằng url

- Đối tượng request của portlet chia 2 loại : renderrequest và actionrequest

Thực sự thì portlet application chỉ là sự mở rộng của web application. Cả hai loại đều được build dưới dạng file war (Web archive file) và dùng Web application deployment descriptor (web.xml). Ngoài ra ứng dụng porlet còn sử dụng thêm 1 file descriptor là portlet.xml để config các portlet.



Sơ đồ tương tác

Tham khảo:

<http://www.vocw.edu.vn/component/content/article/48-ma-nguon-mo-khac/172-tong-quan-ve-portal-liferay-portal.html>

<http://en.wikipedia.org/wiki/Liferay>

<https://sites.google.com/site/nvchuong30/courses/liferay-portal>

**Java Portlet API (JSR 168)**

**1- Những nền tảng của Portlet**

Một server portal quản lý các yêu cầu của client. Giống như một ứng dụng Web phía máy chủ có một web container để quản lý viêc “chạy” các thành phần web ( như servlets, jsps, bộ lọc filters, v.v…), một portal có 1 portlet container để quản lý việc chạy các Portlets. Chú ý rằng hầu hết các ứng dụng web phía máy chủ, chẳng hạn như Tomcat, có thêm các tính năng bổ sung bên cạnh web container ( console quản lý, CSDL người dùng, vvv), còn bao gồm vài ứng dụng web đặc biệt ( chẳng hạn ứng dụng web administration).

Portal được cho là 1 mẫu tương tự, cung cấp ở mức cao hơn các chức năng bao quanh portlet container chúng gắn chặt với đặc tả, cho phép ứng dụng portlet trở nên khả chuyển, giống như ứng dụng web.

Portlet API là 1 mở rộng của đặc tả servlet, điều đó có nghĩa là 1 portlet container cũng vậy, theo định nghĩa 1 web container . Hình 1.1 chỉ ra stack Portal, nó xác nhận làm thế nào các phần khác nhau được xây dựng phía trên nhau để cung cấp 1 portal server.

**Định Nghĩa**

**Portal:**là 1 ứng dụng nền tảng web phía máy chủ chúng tích hợp tuỳ biến và cá nhân hoá nội dung để cho ra một tầng biểu diễn cho hệ thống thông tin xí nghiệp (EIS).

**Portlet:**1 thành phần web có khả năng gắn nối ( plugable) được quản lý bởi 1 portlet container, cái cung cấp một cách linh động nội dung như là một phần của sự kết hợp giao diện người dùng.

**Fragment** Kết quả việc thực thi một portlet, một đoạn các ngôn ngữ đánh dấu (HTML, XML..) nó gắn với một vài qui tắc.

**Portlet Container**  
Môi trường thực thi của 1 portlet. Nó quản lý vòng đời của portlet và quản lý các yêu cầu từ portal bằng cách triệu gọi các portlets bên trong container .

**1.1 – Portlets và Servlets**

Như đã đề cập trước, Portlet API là 1 mở rộng của servlet API. Thế nên, có cả sự tương đồng và sự khác biệt giữa các thành phần. Điều này là quan trọng để hiểu những nét độc đáo (đặc thù) này để hiểu tại sao lai có 1 portlet đặc tả ( và chấm dứt thói quen trang trí bằng cách sử dụng servlet tương tự).

**ĐIỂM TƯƠNG ĐỒNG :**

- Portlet và Servlet đều là thành phần web J2EE.  
– Cả 2 được quản lý bởi container , điều khiển sự tương tác giữa chúng và vòng đời.  
– Mỗi cái sản sinh nội dung web động thông qua cơ chế request/response .

**ĐIỂM KHÁC BIỆT :**

- Portlet sinh ra fragments, trong khi servlets sinh ra một tài kiệu hoàn chỉnh.  
– Không giống servlet, portlet không nhảy tới trực tiếp một URL.  
– Portlet có 1 lược đồ request phức tạp hơn với 2 loại yêu cầu là : action (hành động) – render (đáp ứng).  
– Portlet gắn chặt đến 1 một tập chuẩn hoá các trạng thái, modes chúng định nghĩa các thao tác ngữ cảnh và những qui tắc đáp ứng (render).

**ĐIỂM VƯỢT TRỘI :**

- Portlet có 1 cơ chế phức tạp hơn để truy cập và cố gắng cấu hình thông tin.  
– Portlet phải truy cập đến hiện trạng (profile) thông tin người dùng, ngoại trừ người dùng cơ sở và giữ chức năng cung cấp thông tin trong đặc tả servlet .  
– Portlet có thể thực hiện việc viết lại portlet, vì thế để tạo 1 liên kết thì nó độc lập với việc cài đặt ứng dụng portal server ( nó có rất nhiều phương thức để theo dõi thông tin phiên làm việc…)  
– Portlet có 2 đích sessions khác nhau tron đó lưu trữ các đối tượng : ứng dụng chung và portlet riêng tư.

**ĐIỂM YẾU HƠN :**

- Portlet khong thể thay đổi HTTP header hay thiết lập mã hoá các trả lời (response)  
– Portlet không thể truy xuất URL mà client dùng để khởi tạo các request trên portal.

Các ứng dụng portlet là mở rộng của ứng dụng WEB. Vì thế cả 2 ứng dụng được triển khai (deploy) trong file WAR (Web Archive file) và cả 2 bao gồm 1 file mô tả triển khai ứng dụng web (web.xml). Tuy nhiên 1 ứng dụng portlet còn bao gồm 1 file mô tả triển khai ứng dụng portlet (portlet.xml)

Vì 1 ứng dụng portlet là 1 mở rộng của ứng dụng web, nên logics mà nói nó có thể bao gồm những thành phần ứng dụng web khác. Portlet có thể sử dụng JSPs và servlet để cài đặt những tính năng của nó.

**1.2 – Những tương tác trong Portal (Portal Interactions)**

Thật có ý nghĩa để chỉ ra rằng làm thế nào 1 tương tác portal điển hình xuất hiện trước khi đi vào chi tiết làm thế nào 1 portlet có thể tự hoàn trả (render ) với JSPs và servlet .

Hình 1.2 ở trên chỉ ra 1 chuỗi các sự kiện xuất hiện bên trong portal để quản lý 1 hồi đáp (render ) điển hình của client. Bên trong portal là Portlet API – phục tùng mệnh lệnh của portlet container chúng quản lý trạng thái thực thi của portlet .

Container đánh giá những portlet đó thành các fragments, hoặc là tạo yêu cầu (request ) của portlet hoặc là lấy 1 fragment trong cache. Sau đó, container nắm fragment đến portal server để kết hợp chúng vào trong trang portal.

Nãy giờ chúng ta đã nhìn về portal portlet và portal container ở mức cao hơn, đã đến lúc đào bới vào những đặc tả làm thế nào xây dựng 1 portlet .

**2 – 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 (extend) lớp GenericPortlet để xây dựng portlet, bởi nó cung cấp kiến trúc chứa tất cả những phương thức cài đặt portlet điển hình, không chỉ đơn giản những cái anh cần.

**2.1 – Vòng đời Portlet :**

Rất giống như servlet, vòng đời 1 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 1 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) 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 cho chúng nếu có thể). Tuy nhiên, đáng chú ý là 1 unavailableException có thể xác định thời gian mà portlet sẽ không thích hợp. Cả 2 đều có ích ( giữ cho portlet container luôn cố gắng liên tục tải portlet ) và làm bực mình nhà phát triển ( tại sao portlet container không tải lại portlet của tôi ?).

Phương thức destroy cung cấp 1 cơ may 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 1 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, và phải quản lý chúng trong khối try – catch exception .  
 **2.2 – Các trạng thái thực thi portlet (Portlet Runtime States):**

Khi 1 portlet đang chạy, nó có 1 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 1 sự truy cập đầy đủ một cách hệ thống đến tham chiếu của nó. Khi 1 portlet được đặt vào 1 trang, một Preferences sẽ tham chiếu đến nó. Sự kết đôi của portlet và đối tượng Preferences trên 1 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 1 portlet thư mục làm công (employee directory portlet). Hiển nhiên là, nó cần 1 vài tham chiếu mới có thể chạy được. Tuy nhiên, khi employee directory portlet được nhúng vào trong trang chủ của “Bộ Tài Chính”, nên không chỉ có 1 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 Tài chính.

**2.3 – Quản lý yêu cầu portlet (Portlet Request Handling):**

Có 2 loại yêu cầu (request ) có thể đưa ra đối với 1 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.

**2.3.1 – ” Chỉ có thể là một”**

Nếu yêu cầu của client là action request, thì nó chỉ hướng đến 1 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 .

Hình 1.3 minh hoạ làm thế nào 1 portal container có thể quản lý 1 action request.

Như bạn có thể thấy, 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 1 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 1 user có thể thay đổi chỉ 1 portlet tại 1 thời điểm (bạn chỉ có thể click trên 1 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 qian đã cho.

Hãy xem xét ví dụ chung sau : 1 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 1 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 1 biệt lệ (exception ) để khi 1 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 1 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 1 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 1 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.

2.3.2 – 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

2.3.3 – 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.  
 **2.4 – 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 :