Tổng quan về Portlet

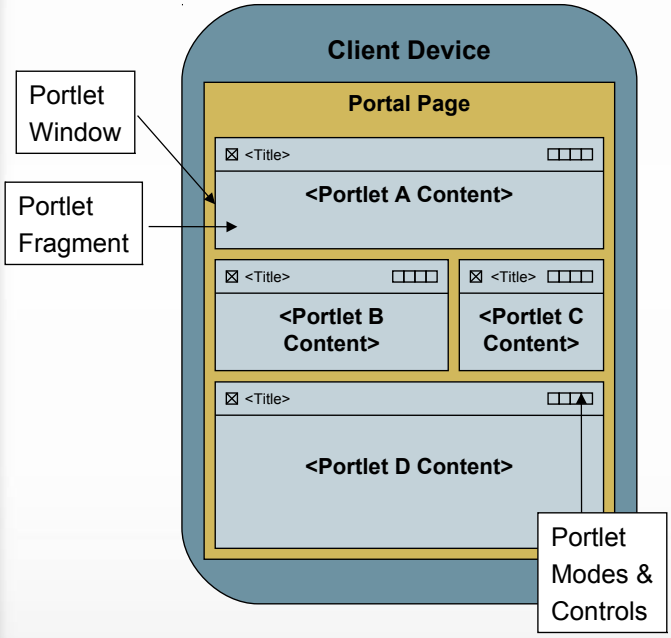
Sở Thông Tin Truyền Thông Vĩnh Long
12/2012

Trình bày: TS. NGÔ BÁ HÙNG
http://ngôbáhùng.vn

Nội dung

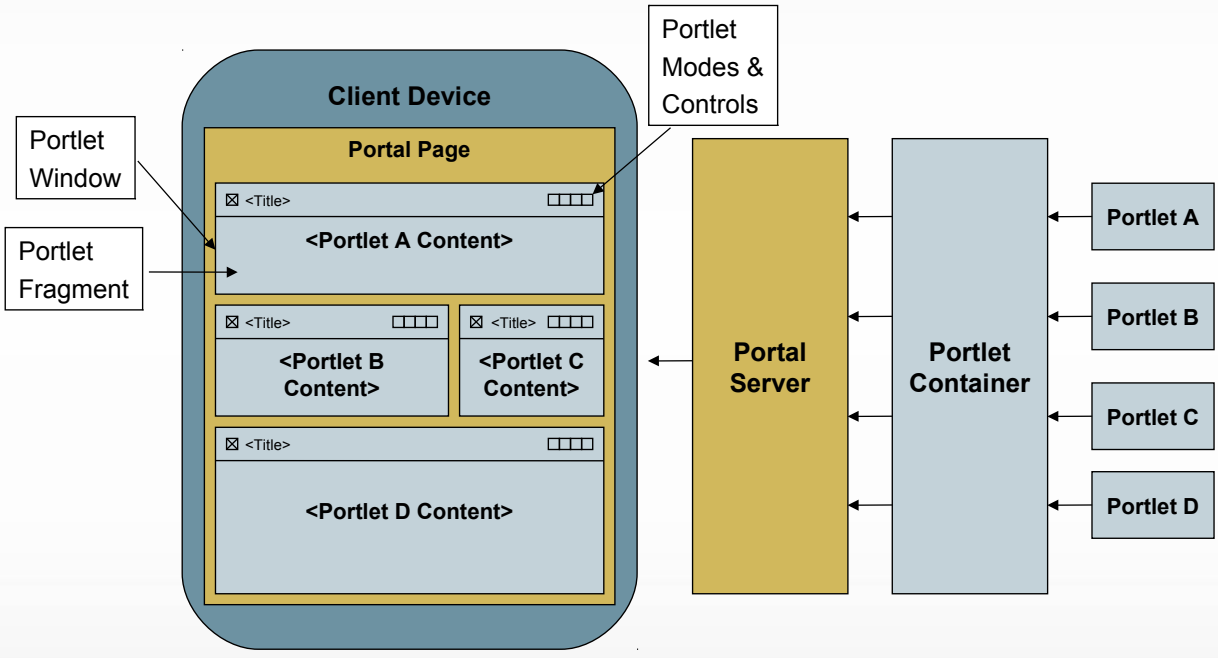
- Portlet là gì
- Dòng đời của portlet
- Portlet và yêu cầu nhiều giai đoạn
- Các chế độ của portlet
- Phát triển HelloWorldPortlet

Portal & Portlet



- Portal: một môi trường dựa trên web mà ở đó tất cả ứng dụng người dùng thực thi tích hợp với nhau một các có hệ thống
- Portet: Là một ứng dụng cung cấp một phần nội dung được nhúng vào như là một phần của một trang thông tin (portal page)

Thành phần của portal



Môt portlet: là một kênh thông tin hoặc một ứng dụng chuyên biệt

Portlet & Servlet (1)

- Các điểm giống với Servlet
  - Portlet và dòng đời của nó được quản lý bởi một container chuyên biệt (Portlet container)
  - Portlet tương tác với web client thông qua một request/response cơ chế
- Các điểm khác với Servlet
  - Portlet chỉ tạo ra một phần của tài liệu HTML, không phải toàn bộ

Portlet & Servlet (2)

- Các điểm khác với Servlet
  - Web client tương tác với portlets qua portal system
  - Portlets có nhiều bộ xử lý yêu cầu hơn: action request, event request, render request
  - Portlet có thể xuất hiện nhiều lần trên một trang
  - Portet có thể lưu dữ liệu trong portlet session ở 2 phạm vi: toàn ứng dụng hoặc trong portlet
  - Portlet không thể thiết đặt bảng mã ký tự trong render response

Portlet container (1)

- Thường nằm phía trên servlet container
- Là một thành phần chức năng của portal server
- Được cài đặt theo đặc tả
  - Java Portlet 1.0 Specifications – JSR168 / 2003
  - Java Portlet 2.0 Specifications – JSR268 / 2008
- Thực thi các portlets và cung cấp một môi trường thực cần thiết cho các portlets
- Cung cấp cơ chế lưu trữ các thông tin tham khảo đến portlet (portlet preferences)

Portlet container (2)

- Nhận các yêu cầu từ portal → Thực thi các yêu cầu trên các portlets đặt trên nó → Tập hợp các trả lời từ portlets và gởi ngược về portal
- Một số Portlet Container & Portal Server phổ biến
  - Apache Pluto / Apache Jetspeed;
  - OpenPortal Portlet Container (Sun Java Portal Server);
  - WebSphere Portlet Container (IBM WebSphere Server);
  - JBoss portlet Container (JBoss Portal),
  - Liferay (Lifeay Portal)

Porlet 1.0 (JSR 168)

- Được phát hành năm 2003 nhằm chuẩn hóa cách thức phát triển các ứng dụng cho các portal server
- Định nghĩa một Portlet API chung và các chủ đề:
  - Portlet container contract & Portlet lifecycle management
  - Packaging anh Deployement
  - Porlet modes & Window states
  - Portlet Preferences Management
  - User Information and Security
  - Localization and Caching
  - JSP tag for Portlet Development

Portlet 2.0 (JSR 286)

- Bổ sung các vấn đề sau
  - Portlet Events
  - Public Render Parameters
  - Resource Serving
  - Portlet filter
  - Caching changes

Dòng đời của Portlet (1)

- Được đặc tả trong JSR 168 gồm 3 giai đoạn
- Khởi tạo: khởi tạo portlet và đưa portlet vào phục vụ
- Xử lý yêu cầu: xử lý các loại yêu cầu khác nhau, có 2 loại yêu cầu:
  - Yêu cầu hành động (action request)
  - Yêu cầu vẽ lại (render request)
- Kết thúc: Đưa portlet về trạng thái không phục vụ
- Được quản lý bởi portlet container thông qua việc gọi các phương thức trên portlet Interface

Dòng đời của Portlet (2)

- Tất cả các portlet phải cài đặt giao diện hoặc kế thừa từ một lớp đã cài đặt giao diện portlet javax.portlet.Portlet
- Các phương thức của giao diện javax.portlet.Portlet
  - init(PortletConfig config): khởi tạo portlet, chỉ gọi 1 lần
  - processAction(ActionRequest request, ActionResponse response): Báo hiệu với portlet rằng người dùng vừa thực hiện một hành động gì đó trên portlet. Phương thức này sẽ chuyển hướng, thay đổi trạng thái portlet, tính toán, thay đổi dữ liệu, đặt các thông số để vẽ lại portlet

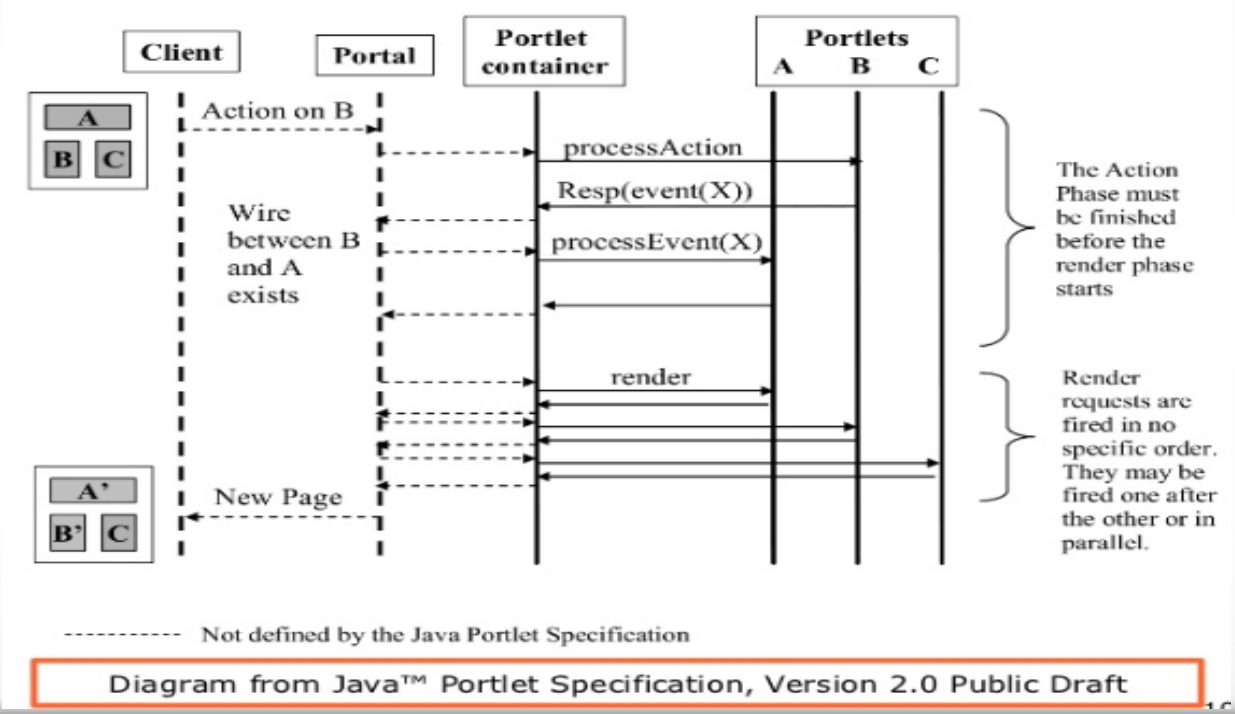
Dòng đời của Portlet (3)

- Các phương thức của giao diện javax.portlet.Portlet
  - render(RenderRequest request, RenderResponse response): Để tạo các thẻ HTML. Các portlet của một trang đều được gọi phương thức render để tạo ra các thẻ HTML mới tùy thuộc vào portlet mode, window state, render parameters, request attributes, persistent state, session data, backend data
  - destroy(): báo hiệu với portlet kết thúc dòng đời của nó. Phương thức này sẽ giải phóng tài nguyền, cập nhật các dữ liệu thuộc về portlet

Yêu cầu nhiều giai đoạn

- Action requests
  - Thực hiện chỉ một lần
  - Được sử dụng để thay đổi trạng thái hệ thống (ví dụ đón nhận và xử lý dữ liệu được post lên từ form)
  - Không tạo ra thẻ HTML
- Render requests
  - Thực thi ít nhất một lần và có thể lặp lại nhiều lần
  - Tạo ra các thẻ HTML
  - Kết quả có thể được trữ lại (cached)

Yêu cầu nhiều giai đoạn (2)



Xử lý action request

- Dùng phương thức processAction(ActionRequest request, ActionResponse response)
  - Được portlet container gọi khi một hành động nào đó được thực hiện trên portlet.
  - ActionRequest chứa đựng thông tin gởi đến bởi người dùng (thông tin nhập trong form)
  - ActionRequest được sử dụng để chuyển sang một trang mới sau khi đã xử lý xong hành động

Xử lý render request (1)

- Yêu cầu loại này sẽ được gởi đến phương thức render(RenderRequest request, RenderResponse response).
- Tuy nhiên không cài đặt code trực tiếp vào render() mà cài vào các phương thức tương ứng với các chế độ của portlet như doView(), doEdit() hay doHelp().
- Phương thức render() sẽ xác định chế độ được yêu cầu bởi người dùng và từ đó gọi phương thức do*Mode*() tương ứng.

Xử lý render request (2)

- Portlet container gởi RenderRequest và RenderResponse đến render()
- Lập trình viên có thể viết các thẻ HTML thông qua đối tượng trả về từ renderResponse.getWriter().
- Tuy nhiên RenderRequest không nhận trực tiếp các giá trị gởi lên từ form mà nhận thông qua đối tượng của lớp PortletSession

Portlet modes (1)

- Một chế độ của portlet (porlet mode) thể hiện chức năng mà một portlet thực hiện
- Các portlets thi hành nhiều tác vụ và tạo ra nội dung khác nhau tùy thuộc vào chức năng mà chúng thực hiện
- Một chế độ báo cho portlet biết tác vụ gì nó phải thực hiện và nội dung gì cần phải tạo ra

Portlet modes (2)

- 3 chế độ cơ bản là: View, Edit và Help
  - View: Hiển thị bình thường của portlet
  - Edit: Thay đổi thông tin tham khảo đến portlet, trạng thái hệ thống
  - Help: Hiển thị tài liệu hướng dẫn về portlet
- Ngoài ra còn có các chế độ tùy biến khác
- Portlet có thể chuyển trạng thái của chính nó

Thông tin tham khảo portlet

- Portlet có thể lưu trữ dữ liệu cho một người dùng bằng cách sử dụng PortletPreferences (thông tin tham khảo portlet)
- Các thông tin tham khảo có thể được đọc và ghi trong giai đoạn action và đọc trong giai đoạn render
- Khi thông tin tham khảo được ghi, portlet ở trong chế độ Edit

Tạo dự án hello-world-portlet (1)

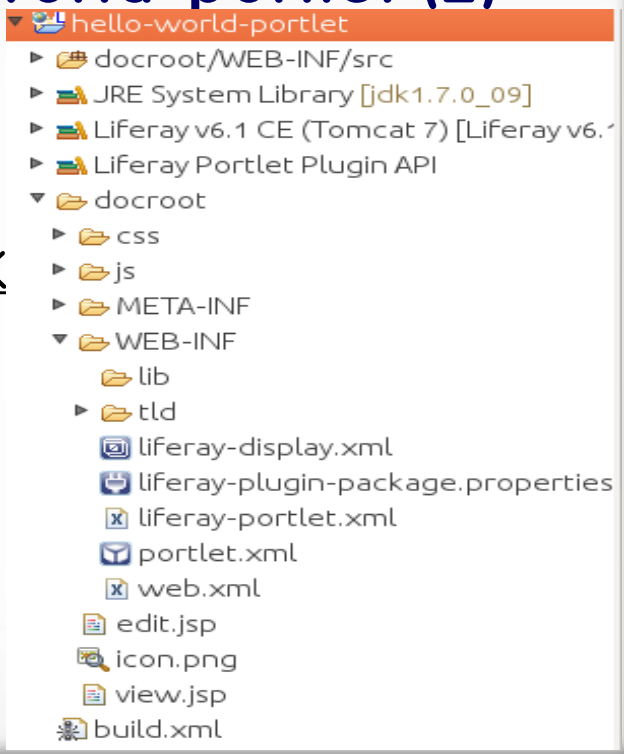
- File → Liferay Project
- Project name: hello-world-portlet
- Display name: Hello World
- Plugin Type: Portlet → Next
- Select portlet framework: Liferay MVC
- Addition Options:
  - Create custom portlet class: CHECK
- Next

Tạo dự án hello-world-portlet (2)

- Portlet class: HelloWorldPortlet
- Java package: vn.edu.ctu
- Superclass: javax.portlet.GenericPortlet
- Next
- Portlet info:
  - Name: hello-world
  - Display name: My Hello World Portlet
  - Title: My Hello World Portlet

Tạo dự án hello-world-portlet (2)

- Portlet modes:
  - Edit: CHECK
- Create JSP files: UN CHECK
- Next → Next
- processAction: CHECK
- Finish



portlet.xml

```
<portlet>
    <portlet-name>hello-world</portlet-name>
    <display-name>My Hello World Portlet</display-name>
    <portlet-class>vn.edu.ctu.HelloWorldPortlet</portlet-class>
    <init-param>
        <name>view-template</name>
        <value>/view.jsp</value>
    </init-param>
    <init-param>
        <name>edit-template</name>
        <value>/edit.jsp</value>
    </init-param>
    <expiration-cache>0</expiration-cache>
    <supports>
        <mime-type>text/html</mime-type>
        <portlet-mode>view</portlet-mode>
        <portlet-mode>edit</portlet-mode>
    </supports>
```


HelloWorldPortlet – init()

- public class HelloWorldPortlet extends GenericPortlet {

```
    public void init() {
        // Đọc các giá trị các phần tử trong phần <init-param>
        // của tập tin portlet.xml
        editJSP = getInitParameter("edit-template");
        viewJSP = getInitParameter("view-template");
    }

....

}
```

HelloWorldPortlet -doView(1)

- public void doView(RenderRequest renderRequest, RenderResponse renderResponse) throws IOException, PortletException {

```
// Thêm mã lệnh ở slide sau vào đây
// ....
//Chuyển điều khiển về tập tin view.jsp
include(viewJSP, renderRequest,
renderResponse);
```

}

HelloWorldPortlet -doView(2)

- Thêm vào doView()

```
// Lấy đối tượng lưu thông tin tham khảo portlet
PortletPreferences prefs =renderRequest.getPreferences();
//Lấy giá trị của thuộc tính có tên là name
// nếu không có thuộc tính name thì trả về giá trị la no
String username = (String) prefs.getValue("name", "no");
// Gán lại giá trị rỗng cho username nếu username =no
if (username.equalsIgnoreCase("no")) {
        username = "";
}
```

- Thêm sau package: import javax.portlet.PortletPreferences;

HelloWorldPortlet - include()

- Phương thức include chuyển điều khiển về một trang mới ở địa chỉ path

```
protected void include(String path, RenderRequest renderRequest,
        RenderResponse renderResponse)

    throws IOException, PortletException {

    PortletRequestDispatcher portletRequestDispatcher =
        getPortletContext().getRequestDispatcher(path);

    if (portletRequestDispatcher == null) {
        _log.error(path + " is not a valid include");
    }
    else {
        portletRequestDispatcher.include(renderRequest, renderResponse);
    }
}
```


view.jsp

- Biên soạn nội dung view.jsp như sau

```
<!--Thư viện tag của portlet -->
<%@ taglib uri="http://java.sun.com/portlet" prefix="portlet" %>
<!--Lấy giá trị của thuộc tính có tên là userName từ đối tượng
request  -->
<jsp:useBean id="userName" class="java.lang.String"
scope="request" />
<!-- Tạo các đối tượng renderRequest renderResponse và
portletConfig cho trang-->
<portlet:defineObjects />
<p>This is the Hello You portlet.</p>
<p>Hello <%= userName %>!</p>
```

HelloWorldPortlet-doEdit(1)

- public void doEdit(RenderRequest renderRequest, RenderResponse renderResponse) throws IOException, PortletException {

```
// Thêm mã lệnh ở slide sau vào đây
// ....
//Chuyển điều khiển về tập tin edit.jsp//
include(editJSP, renderRequest, renderResponse);
}
```


HelloWorldPortlet-doEdit(1)

- Thêm vào doEdit()

```
//Tạo một ActionURL
PortletURL addNameURL =
renderResponse.createActionURL();
//Thêm tham số có tên là addName có giá trị là
addName vào ActionURL
addNameURL.setParameter("addName", "addName");
// Đưa thuộc tính addNameURL vào request để chuyển
giá trị của nó qua edit.jsp
renderRequest.setAttribute("addNameURL",
addNameURL.toString());
```

edit.jsp (1)

- <%@ taglib uri="http://java.sun.com/portlet" prefix="portlet" %>
  <!-- Nhận giá trị của addNameURL từ request -->
  <jsp:useBean class="java.lang.String" id="addNameURL" scope="request" />
  <portlet:defineObjects />
  <form id="<portlet:namespace />helloForm" action="<%= addNameURL %>" method="post"> <!-- form được submit đến addNameURL là một ActionURL sẽ kích hoạt phương thức processAction() để xử lý dữ liệu của form -->

edit.jsp (2)

```
<table>
<tr>
   <td>Name:</td>
   <td><input type="text" name="username"></td>
</tr>
</table>
<input type="submit" id="nameButton" title="Add Name"
value="Add Name">
</form>
```

HelloWorldPorlet - processAction

- public void processAction(ActionRequest actionRequest, ActionResponse actionResponse) throws IOException, PortletException {

```
// Thêm phần đón nhận dữ liệu từ form ở slide sau
//...
// Bỏ qua phương thức processAction của lớp cha
//super.processAction(actionRequest, actionResponse);
```

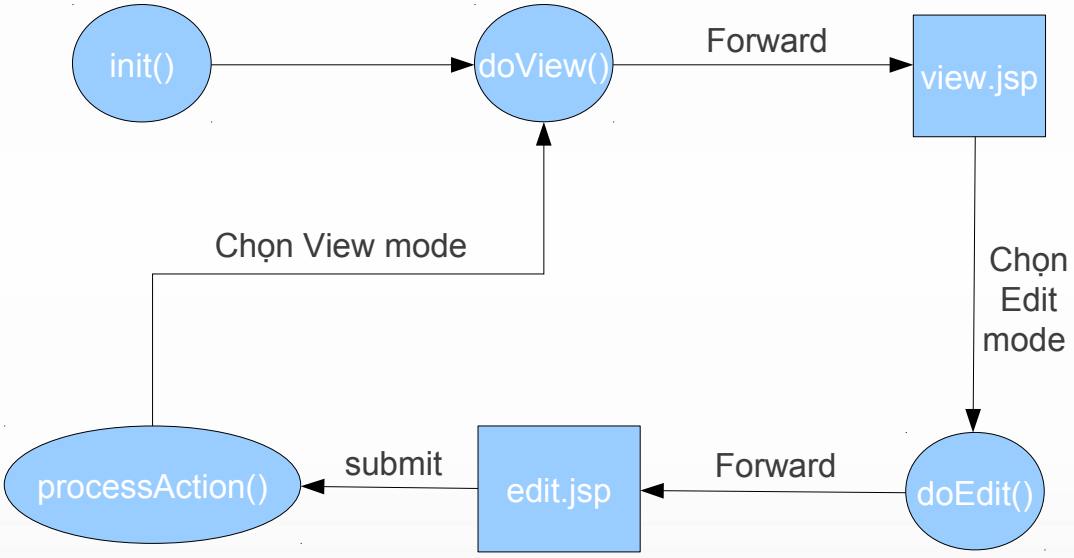
}

HelloWorldPorlet - processAction

- Thêm vào processAction:

```
//lấy giá trị của thuộc tính addName
String addName = actionRequest.getParameter("addName");
if (addName != null) { // Nếu tồn tại thuộc tính
    // Lấy đối thượng thông tin tham khảo portlet
    PortletPreferences prefs =actionRequest.getPreferences();
    // Lưu giá trị của thuộc tính username vào thông tin tham khảo portlet
    prefs.setValue("name",actionRequest.getParameter("username"));
    prefs.store();
    //Đặt portlet vào chế độ view → view.jsp sẽ được load lên
    actionResponse.setPortletMode(PortletMode.VIEW);
}
```

Giao tiếp giữa các thành phần



Triển khai hello-world-portlet