
KIVER
Portlet JSR168/286
Sassari, 21 Maggio 2011

Agenda

1. JSR168
2. JSR286
3. Spring MVC Portlet
4. Esempio pratico

Portale

Portale

Porta di accesso unica a un insieme di applicazioni, dati e servizi.

Applicazioni

- facilmente inseribili nella pagina
- unità indipendenti
- configurabili
- riusabili anche in altri contesti e portali

Portlet - evoluzione

JSR 168 (Java Portlet Specification 1.0)

La specifica JSR 168 è stata rilasciata nel 2003

- al gruppo di lavoro hanno partecipato Apache, BEA, Broadvision, IBM, Oracle, SAP, SUN, Sybase, TIBCO

Definisce

- il ruolo e le funzionalità del Portlet Container
- il contratto tra il Container e i Portlet
- la gestione del ciclo di vita dei Portlet
- il packaging per la distribuzione dei Portlet
- l'interazione con Web Services for Remote Portlets (WSRP) di OASIS

Portlet – Portlet container

Portlet

Componenti web Java, gestiti da un **portlet container**, che processano le richieste ricevute e generano contenuti dinamici.

Portlet container

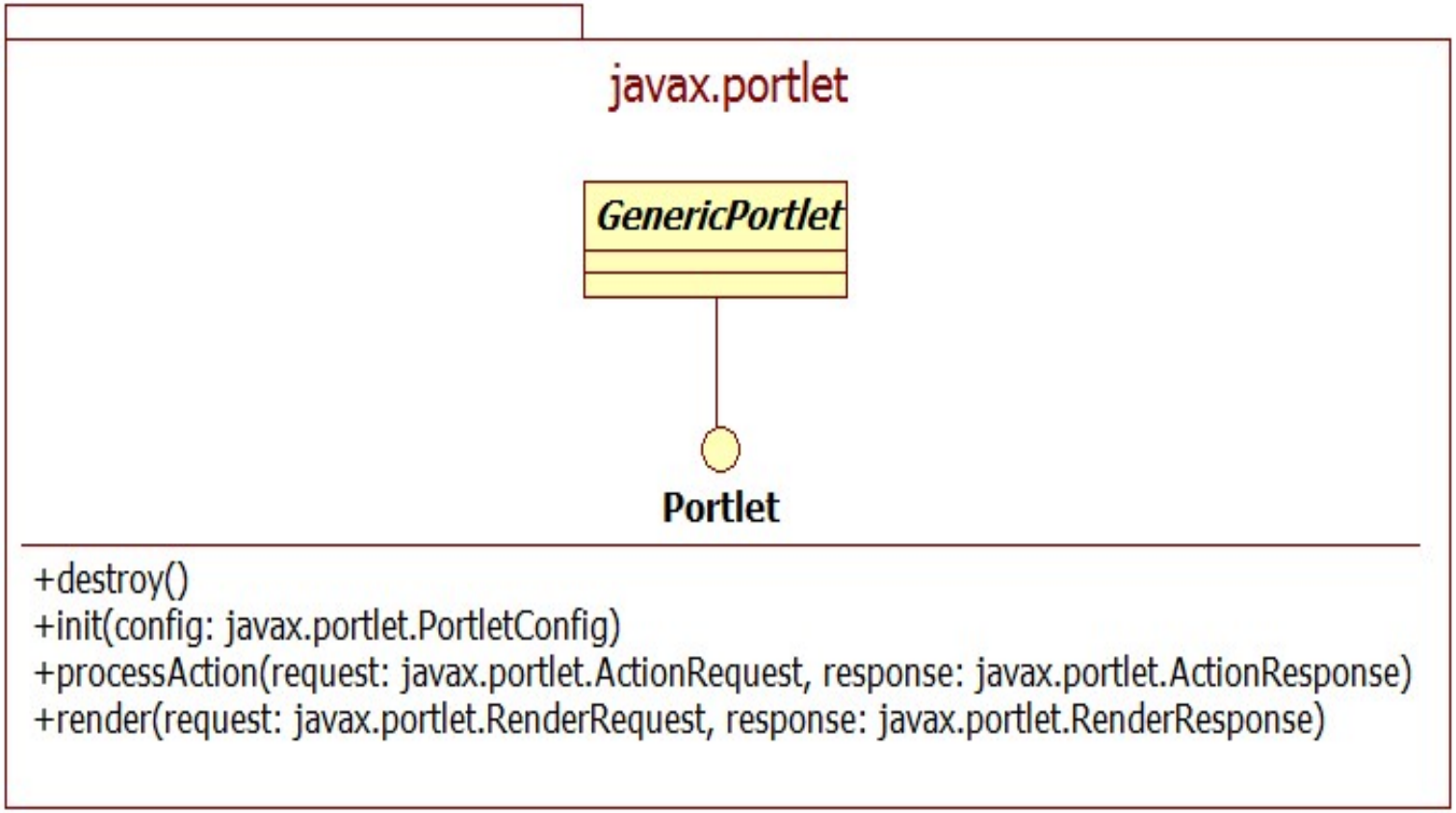
- contiene i portlet e gestisce il loro ciclo di vita.
- si occupa di immagazzinare le preference e le configurazioni di ogni portlet
- riceve le request dal portale e le indirizza alle portlet che gestisce

Ciclo di vita di un portlet

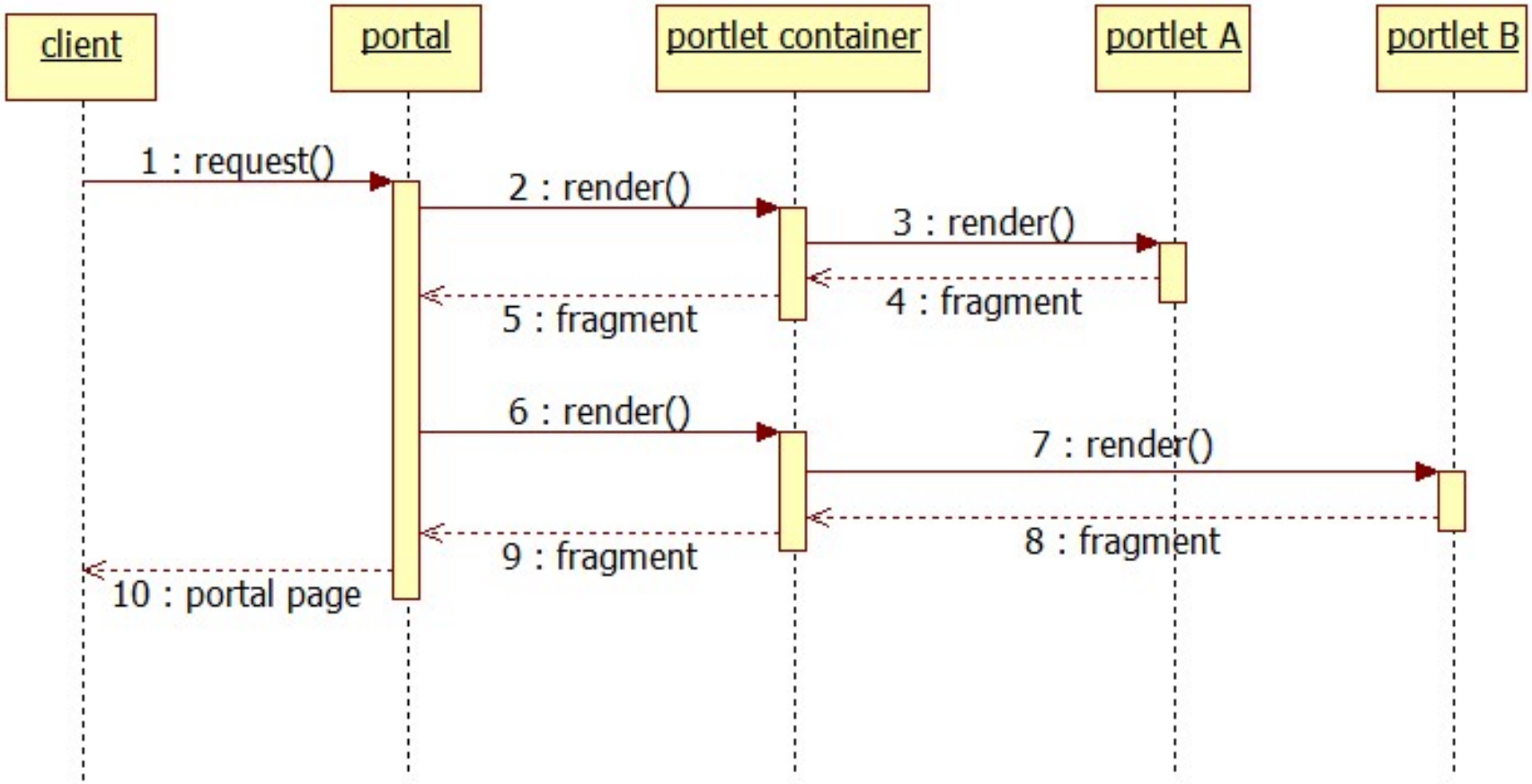
Il ciclo di vita è gestito dai 4 metodi esposti dall'interfaccia portlet:

- **init:** il metodo viene chiamato dal container quando il portlet viene instanziato
- **destroy:** viene chiamato dal container quando il portlet viene distrutto
- **processAction:** viene chiamato dopo che l'utente ha effettuato una richiesta; serve a processare i dati avuti in input
- **render:** va in esecuzione ogni volta che il portlet si visualizza nella pagina web

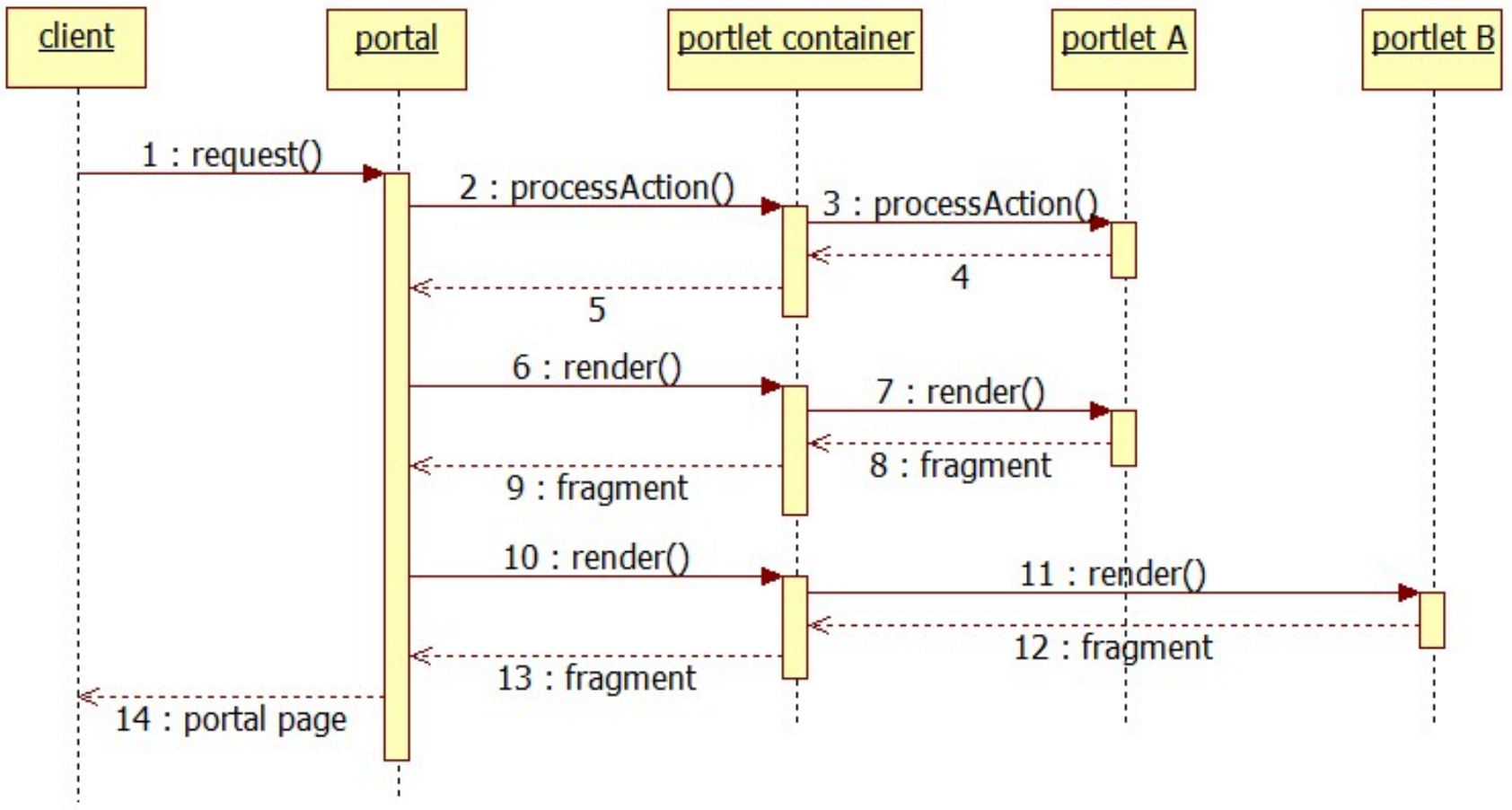
javax.portlet.Portlet



Caricamento pagina



Process di una action



ActionRequest – RenderRequest

- Derivano dall'oggetto PortletRequest
- Sono *sottoinsiemi* dell'HttpRequest
  - è compito del portlet container *dividere* i parametri tra i vari portlet della pagina
- I parametri in actionRequest sono in generale diversi da quelli in renderRequest
  - a meno che il portlet non li inserisca in actionResponse

Esempio

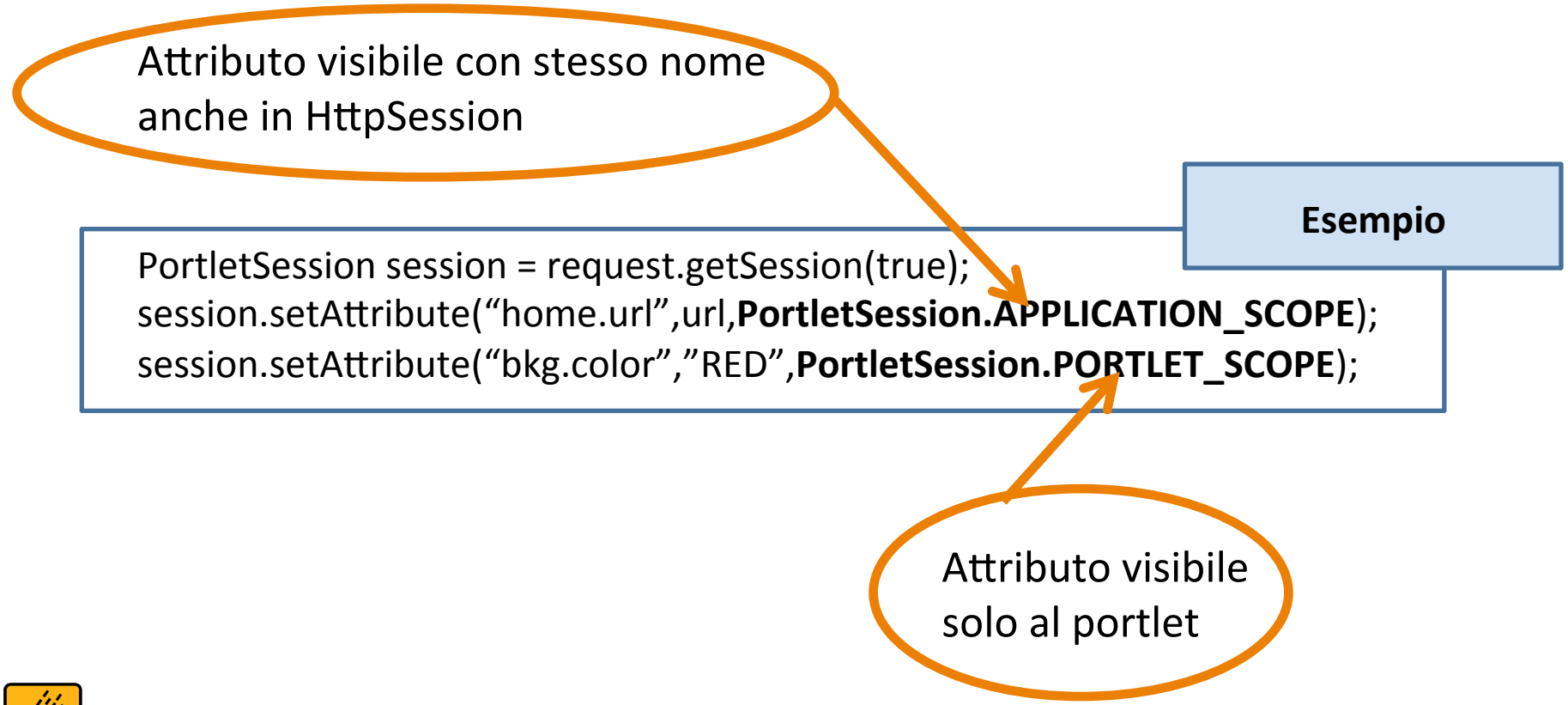
```
String value=request.getParameter("test");
response.setRenderParameter("test", value);
```

Il parametro test è preso dalla actionRequest e passato in actionResponse

PortletSession

Ogni portlet ha il proprio PortletSession

- i parametri inseriti nel PortletSession con lo scope APPLICATION sono visibili anche in HttpSession



Modalità di un portlet

La specifica definisce 3 modalità standard:

- **VIEW:** la modalità standard che ogni portlet deve implementare
- **EDIT:** consente di modificare le impostazioni dell'utente
- **HELP:** tale modalità presenta informazioni di help relative alla portlet

I vari vendor possono comunque aggiungere modalità differenti.

Portlet taglib

URI: http://java.sun.com/portlet

Permettono alle JSP incluse nel portlet

- di avere accesso ad elementi specifici del portlet (render, request, id del portlet)
- di creare i link per l'interazione con il portlet

L'implementazione delle taglib è responsabilità del portlet container

<portlet:namespace/>

Definizione

Ritorna l'identificativo univoco del portlet all'interno della pagina web

Utile per l'assegnazione di id univoci agli elementi della pagina web, soprattutto se sono presenti più istanze dello stesso portlet nella stessa pagina

Esempio

```
var <portlet:namespace/>_resTab=new ResultsTable(..);
 <portlet:namespace/>_populateTable(output);
 </script>
      <table id="<portlet:namespace/>_results">
            <th><td>Nome</td><td>Cognome</td></th>
       </table>
```

<portlet:defineObjects/>

Definizione

definisce le seguenti variabili all'interno della JSP:

- renderRequest
- renderResponse
- portletConfig

Esempio

```
<portlet:defineObjects/>
<%=renderResponse.setTitle("my portlet title")%>
```

<portlet:actionURL/>

Definizione

crea un URL che punta al portlet corrente ed invia un action request con i parametri specificati

Parametri

portletMode: la modalità del portlet
var: il nome della variabile nella JSP

Esempio

```
<portlet:actionURL var="searchAction">
  <portlet:param name="action" value="search" />
</portlet:actionURL>
<form action="${searchAction}" method="POST">
```

<portlet:renderURL/>

Definizione

crea un URL che punta al portlet corrente ed invia un render request con i parametri specificati

Parametri

portletMode: la modalità del portlet
var: il nome della variabile nella JSP

Esempio

```
<portlet:renderURL var="helpUrl" portletMode="HELP">
</portlet:renderURL>
<a href="${helpUrl}">Help</a>
```

Portlet library

I portlet possono essere inserite in un opportuno file war

- il file deve contenere le classi e tutte le resources relative ad essi
- nel folder WEB-INF deve essere presente il file **portlet.xml**

portlet.xml

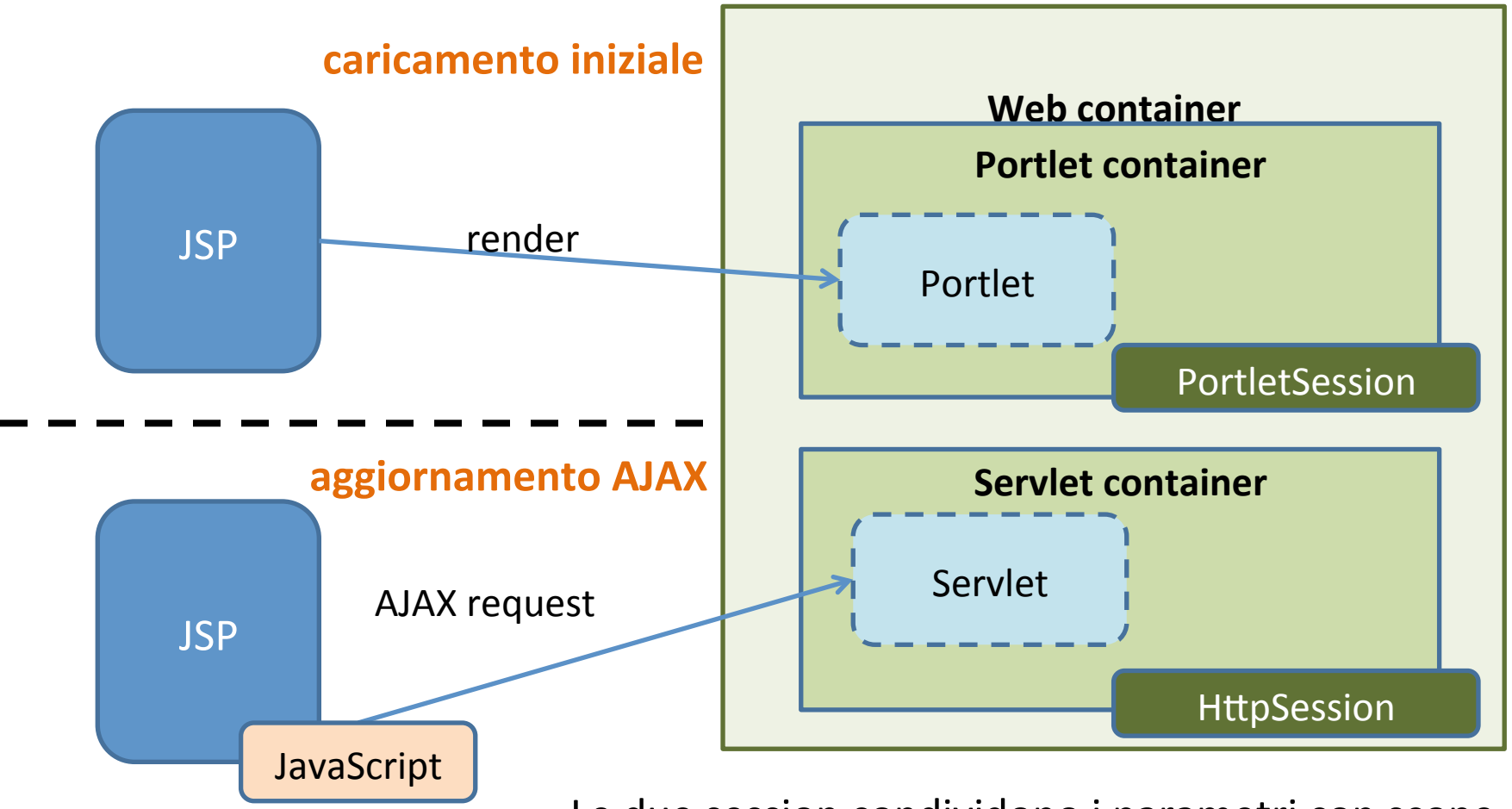
```
<portlet>
  <description xml:lang="it">Portlet di test A</description>
 <portlet-name>TestAPortlet</portlet-name>
<display-name xml:lang="it">Portlet di test A</display-name>
<portlet-class>org.springframework.web.portlet.DispatcherPortlet</portlet-class>
 <init-param>
     <name>contextConfigLocation</name>
     <value>/WEB-INF/context/TestPortletA-context.xml</value>
  </init-param>
  <expiration-cache>0</expiration-cache>
  <supports>
   <mime-type>text/html</mime-type>
   <portlet-mode>view</portlet-mode>
   <portlet-mode>help</portlet-mode>
  </supports>
 <supported-locale>en</supported-locale>
…
</portlet>
```

Limiti di JSR168

La specifica presenta alcune limitazioni:

- manca un meccanismo standard per fare interagire tra loro i portlet (**Inter Portlet Communication**)
- non è possibile ottenere una risorsa direttamente da un portlet
  - bisogna passare dal portlet containet
- integrazione con AJAX non supportata internamente
  - a meno che non sia offerta dal portlet container in modo non standard

Integrazione con AJAX



Le due session condividono i parametri con scope **APPLICATION_SCOPE**

Agenda

1. JSR168
2. JSR286
3. Spring MVC Portlet
4. Esempio pratico

JSR286 (Java Portlet Specification 2.0)

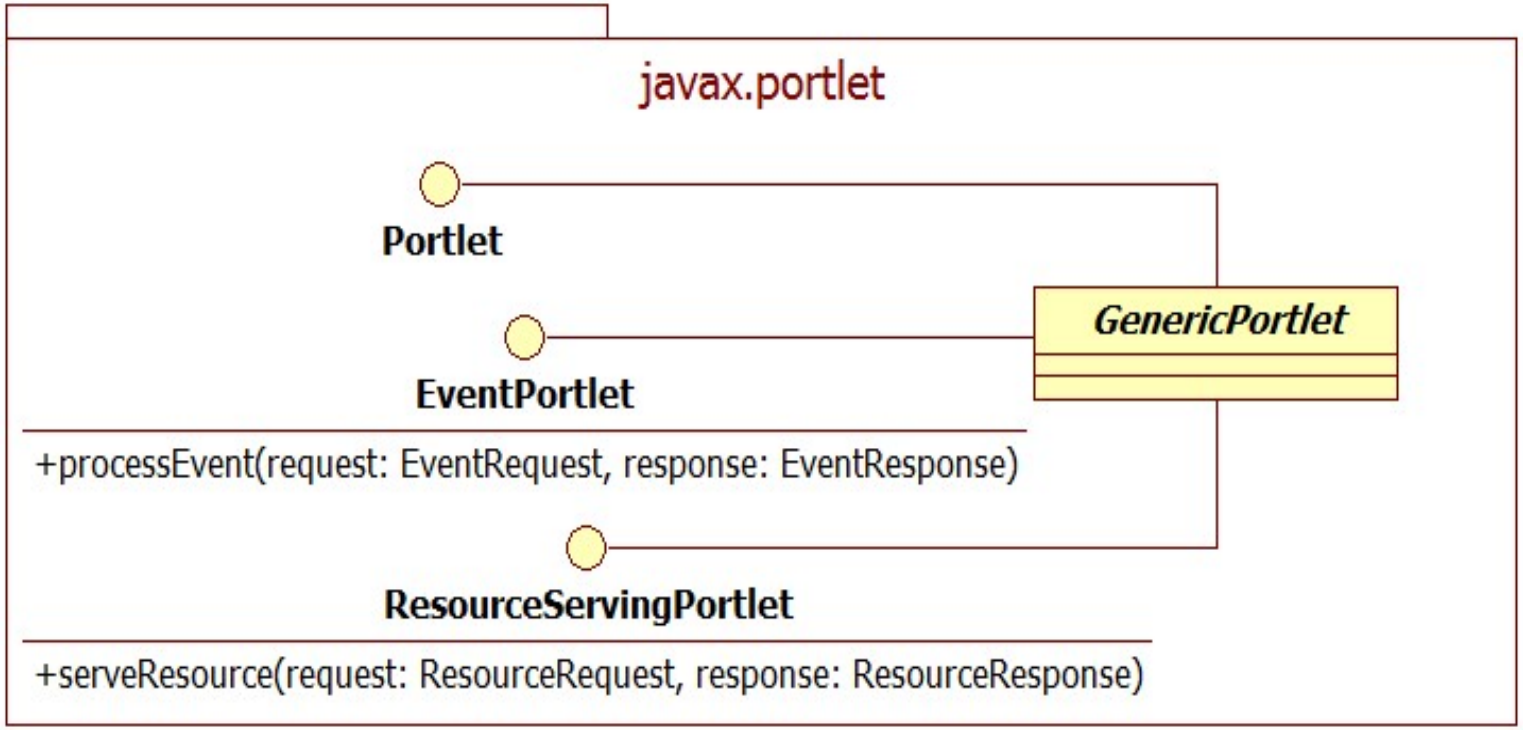
La specifica JSR 286 è stata rilasciata nel 2008

- a febbraio 2006 fu costituito il JSR 286 Expert Group al fine di arrivare alla Java Portlet Specification 2.0

Novità introdotte

- Eventi: ogni portlet può lanciare e ricevere determinati eventi
- Public render parameter: possibilità per i portlet di condividere parametri tra loro
- Possibilità per un portlet di restituire una risorsa.

GenericPortlet



Eventi- dichiarazione dell'evento

portlet.xml

```
<event-definition>
  <qname xmlns:x="http://com.alex/test/portlets/ns">x:event</qname>
  <value-type>java.lang.String</value-type>
</event-definition>
```

Il parametro **value-type** indica la classe dell'evento

Eventi - lancio dell'evento

portlet.xml

```
<portlet>
  <description xml:lang="it">Portlet di test A</description>
  <supported-publishing-event>
      <qname xmlns:x="http://com.alex/test/portlets/ns">x:event</
qname>
  </supported-publishing-event>
```

codice

```
String event="Test Event";
QName name=new QName("http://com.alex/test/portlets/ns", "event");
 response.setEvent(name, event);  //ActionResponse
```

Eventi - process dell'evento

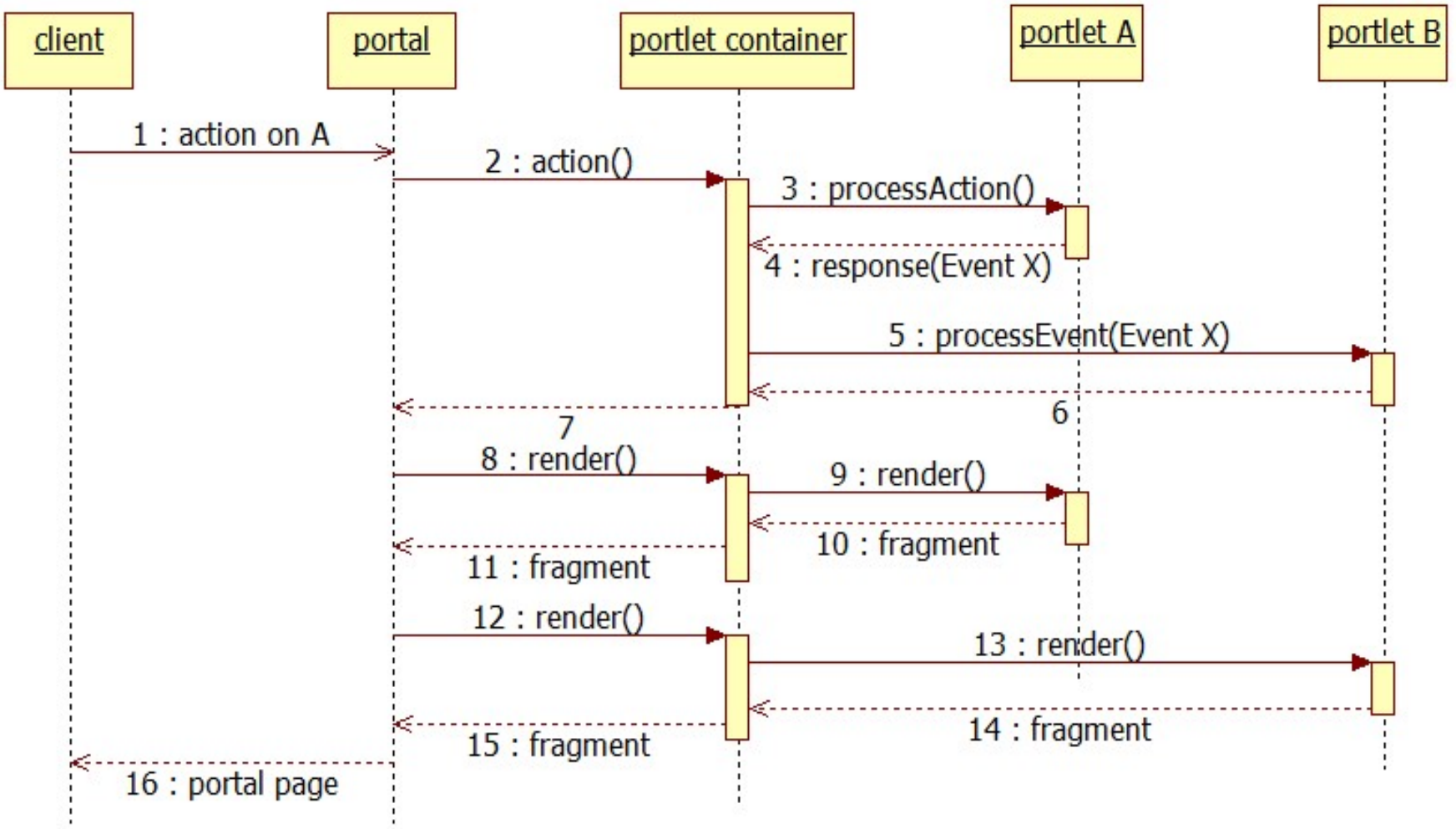
portlet.xml

```
<portlet>
  <description xml:lang="it">Portlet di test B</description>
  <supported-processing-event>
      <qname xmlns:x="http://com.alex/test/portlets/ns">x:event</
qname>
  </supported-processing-event>
```

codice

```
public void processEvent(EventRequest request, EventResponse
response){
   Event event=request.getEvent();
   String qname= event.getQName();
}
```

Eventi - flusso



Parametri pubblici - dichiarazione

I parametri pubblici vengono dichiarati in portlet.xml in una opportuna sezione

- vengono distinti tra loro utilizzando opportuni identifier

portlet.xml

```
<public-render-parameter>
  <identifier>foo</identifier>
  <qname xmlns:x="http://com.alex/test/portlets/ns">x:foo2</qname>
</public-render-parameter>
```

Parametri pubblici - utilizzo

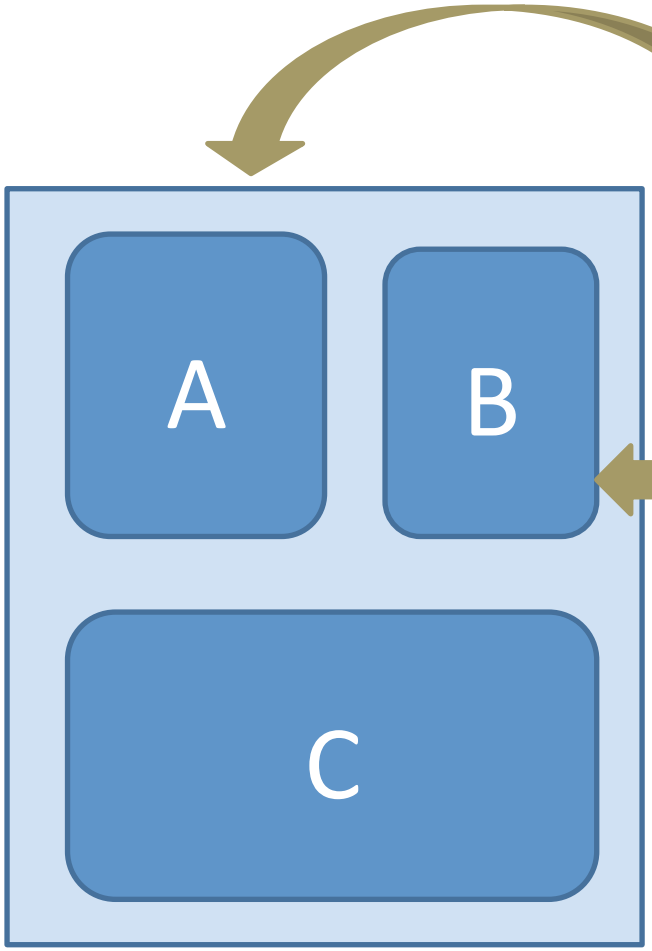
I portlet dichiarano in portlet.xml i parametri pubblici che utilizzeranno

- i parametri sono identificati con il loro identifier

portlet.xml

```
<portlet>
  <portlet-name>portletA</portlet-name>
  …
  <supported-public-render-parameter>foo</supported-public-render-parameter>
</portlet>
```

Parametri pubblici – flusso



Portlet A

```
public void processAction(ActionRequest
request, ActionResponse response){
response.setRenderParameter(“foo", foo);}
```

Portlet B

```
public void render(RenderRequest request,
RenderResponse response){
String value = response.getParameter(“foo");}
```

Resource - <portlet:resourceURL/>

Definizione

crea l'url per una risorsa, identificata da un id, fornita dal portlet

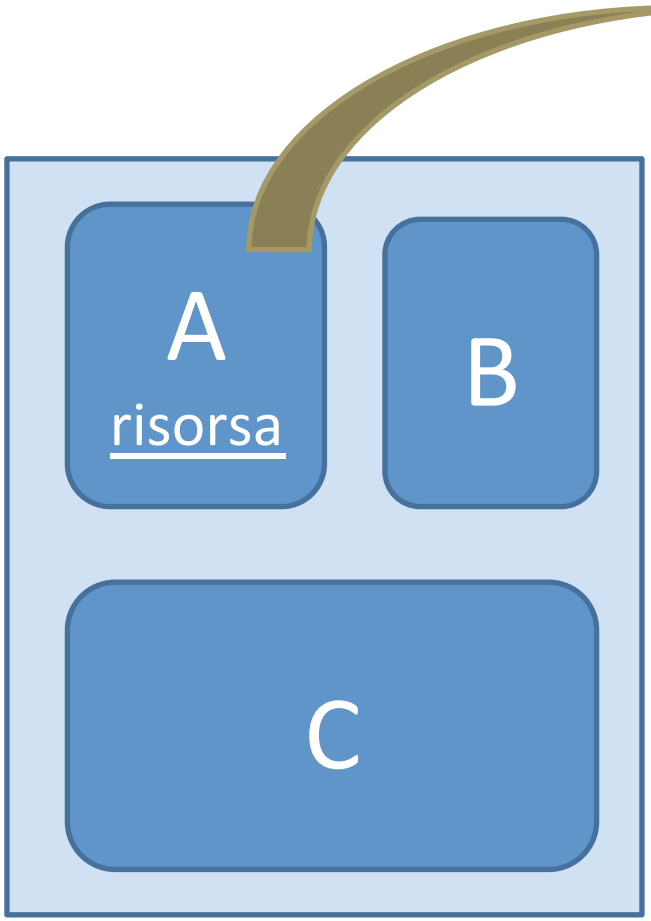
Parametri

var: il nome della variabile all'interno della JSP.
id: l'identificativo della risorsa richiesta

Esempio

```
<portlet:resourceURL var="changePage" id="changePage">
  <portlet:param name="par1" value="val1"/>
</portlet:resourceURL>
```

Resource - flusso

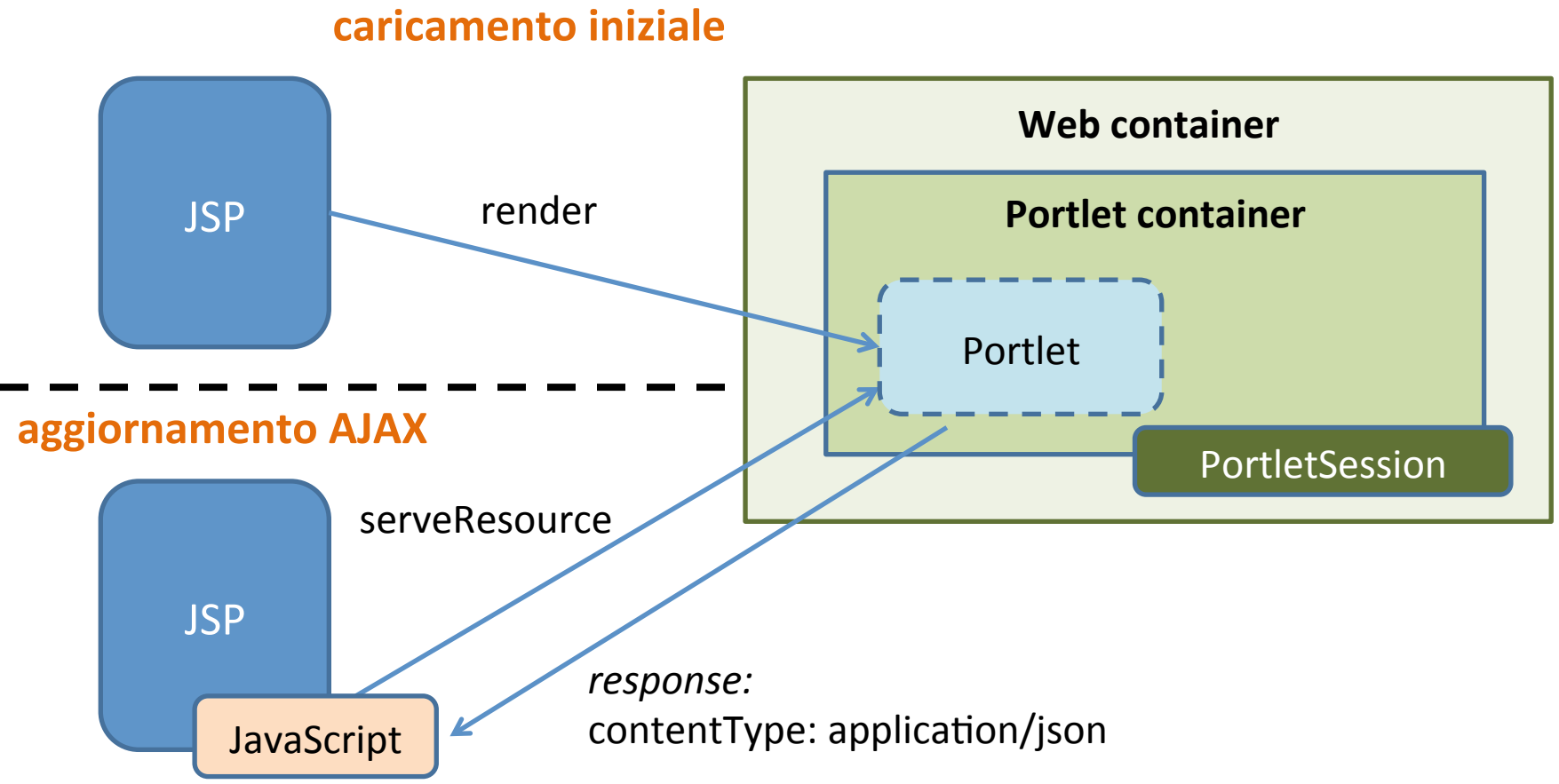



```
public class TestAPortlet extends
GenericPortlet{

    public void serveResource
(ResourceRequest request,
ResourceResponse response){

}
}
```

Integrazione con AJAX



Agenda

1. JSR168
2. JSR286
3. Spring MVC Portlet
4. Esempio

Spring MCV Portlet

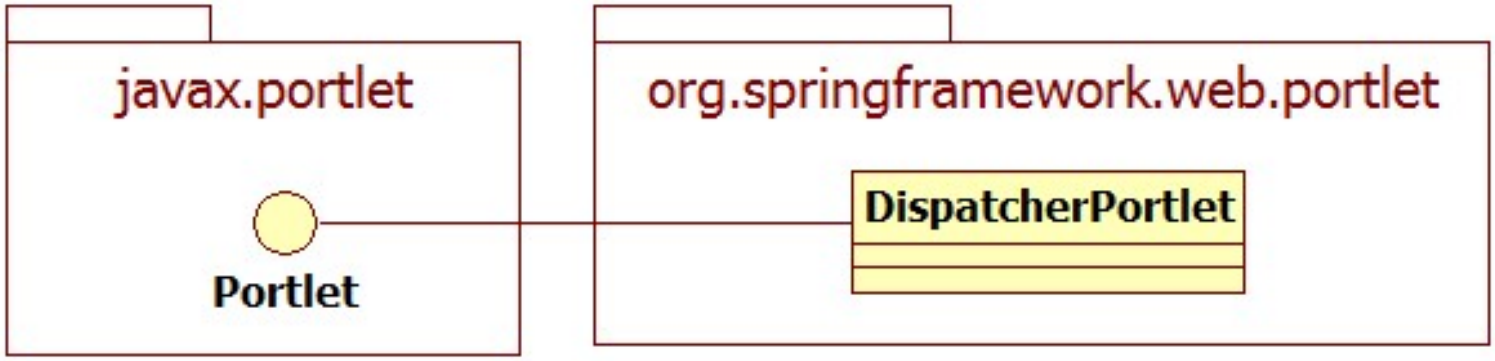
Caratteristiche

- modello flessibile e leggero
- Implementa il classico pattern MVC
- analogo al modello Spring MVC

Componenti

- DispatcherPortlet
- HandlerMapping
- Controller
- ViewResolver

DispatcherPortlet



portlet.xml

```
<portlet-class>org.springframework.web.portlet.DispatcherPortlet</portlet-class>
 <init-param>
     <name>contextConfigLocation</name>
     <value>/WEB-INF/context/TestPortletA-context.xml</value>
  </init-param>
```

HandlerMapping

Responsabilità

Mappa le PortletRequest verso gli opportuni controllers

context.xml

```
<bean
class="org.springframework.web.portlet.mvc.annotation.DefaultAnnotatio
nHandlerMapping">
    <property name="interceptors">
      <bean
class="org.springframework.web.portlet.handler.ParameterMappingInterc
eptor" />
    </property>
</bean>
```

ViewResolver

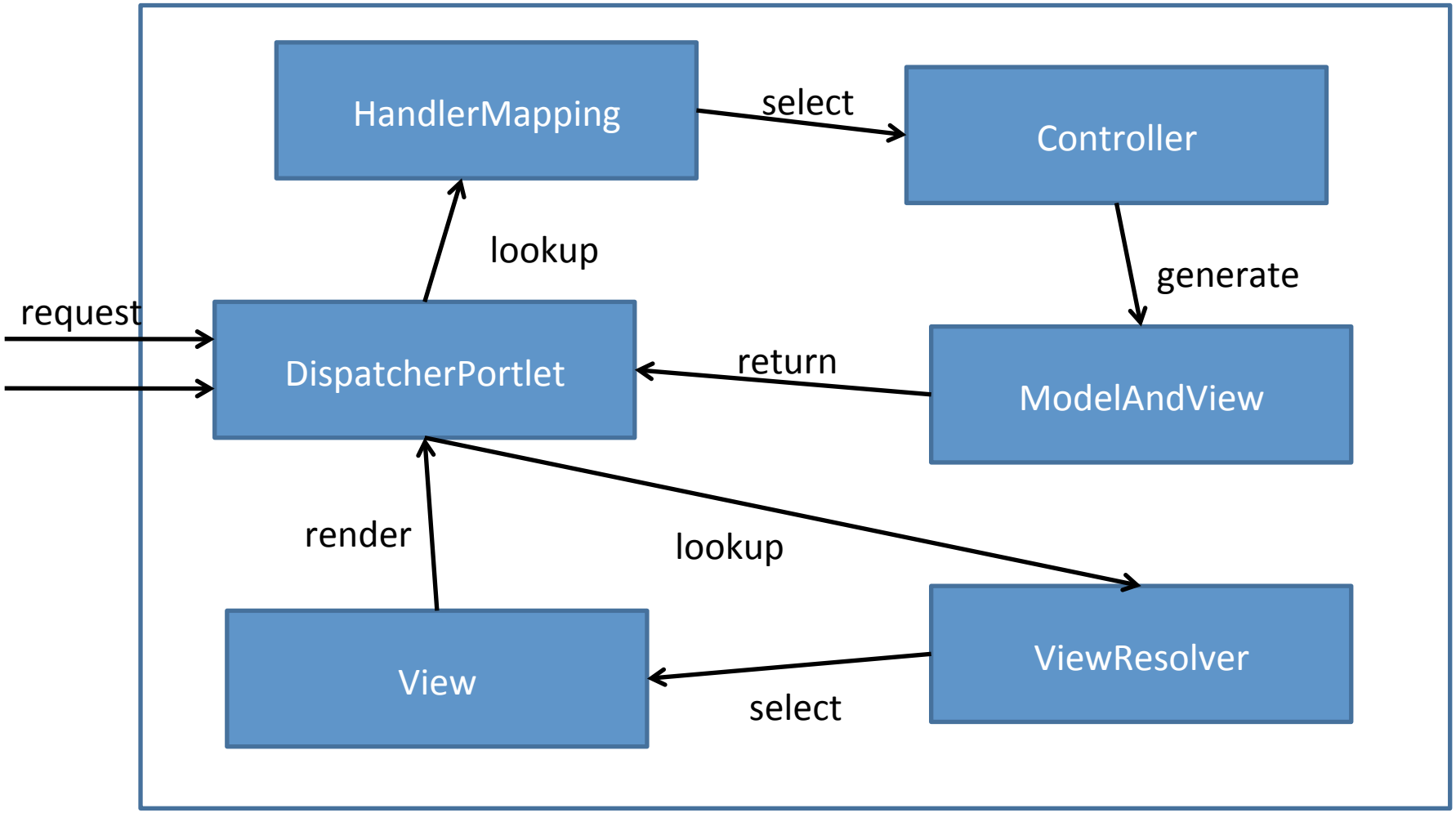
Responsabilità

Sceglie la view opportuna a seconda del contesto

context.xml

```
<bean id="viewResolver"
class="org.springframework.web.servlet.view.InternalResourceView
Resolver">
        <property name="cache" value="true" />
        <property name="viewClass“
value="org.springframework.web.servlet.view.JstlView" />
        <property name="prefix" value="/WEB-INF/jsp/testA/" />
       <property name="suffix" value=".jsp" />
</bean>
```

Interazione tra i componenti



Controller annotati

Nuovi in Spring 2.5

Caratteristiche

- Rendono più snelle le configurazioni
- Permettono di riunire la logica nella stessa classe di controller
- Possibilità di utilizzare segnature 'libere' nei metodi.

Controller annotati – metodi

Parametri metodi

Possono essere in qualsiasi ordine:

- Request/response
- Model o oggetti del model
- Errors
- java.util.Locale

Output metodi

- oggetto ModelAndView
- Model o oggetti del model
- oggetto View
- string che identifica l'oggetto view

esempio

```
public void searchAction(@ModelAttribute("search") SearchUserForm
search,Model model,ActionRequest request){
```

Controller annotati – component-scan

Il tag component-scan indica al framework dove cercare i controller annotati da usare per il portlet

context.xml

```
<context:component-scan base-package=
"com.alex.springportlet.testA" />
        <context:annotation-config />
```

@Controller

La classe marcata con l'annotation **@Controller** viene utilizzata come controller

- il parametro value indica l'id del bean nel file di context

codice

```
@Controller(value="testAController")
public class TestAPortletController extends GeneralController{
```

context.xml

```
 <bean id="testAController"
class="com.alex.springportlet.testA.controller.TestAPortletCo
ntroller">
        <property name="service" ref="serviceImpl"/>…
```

@RequestMapping - 1

A livello di classe,viene usata per distinguere I vari controller in base alle modalità che gestiscono

codice

```
@Controller(value="testAController")
@RequestMapping(“VIEW”)
public class TestAPortletController extends GeneralController{
```

In questo caso il controller
è utilizzato
per gestire la modalità VIEW

@RequestMapping - 2

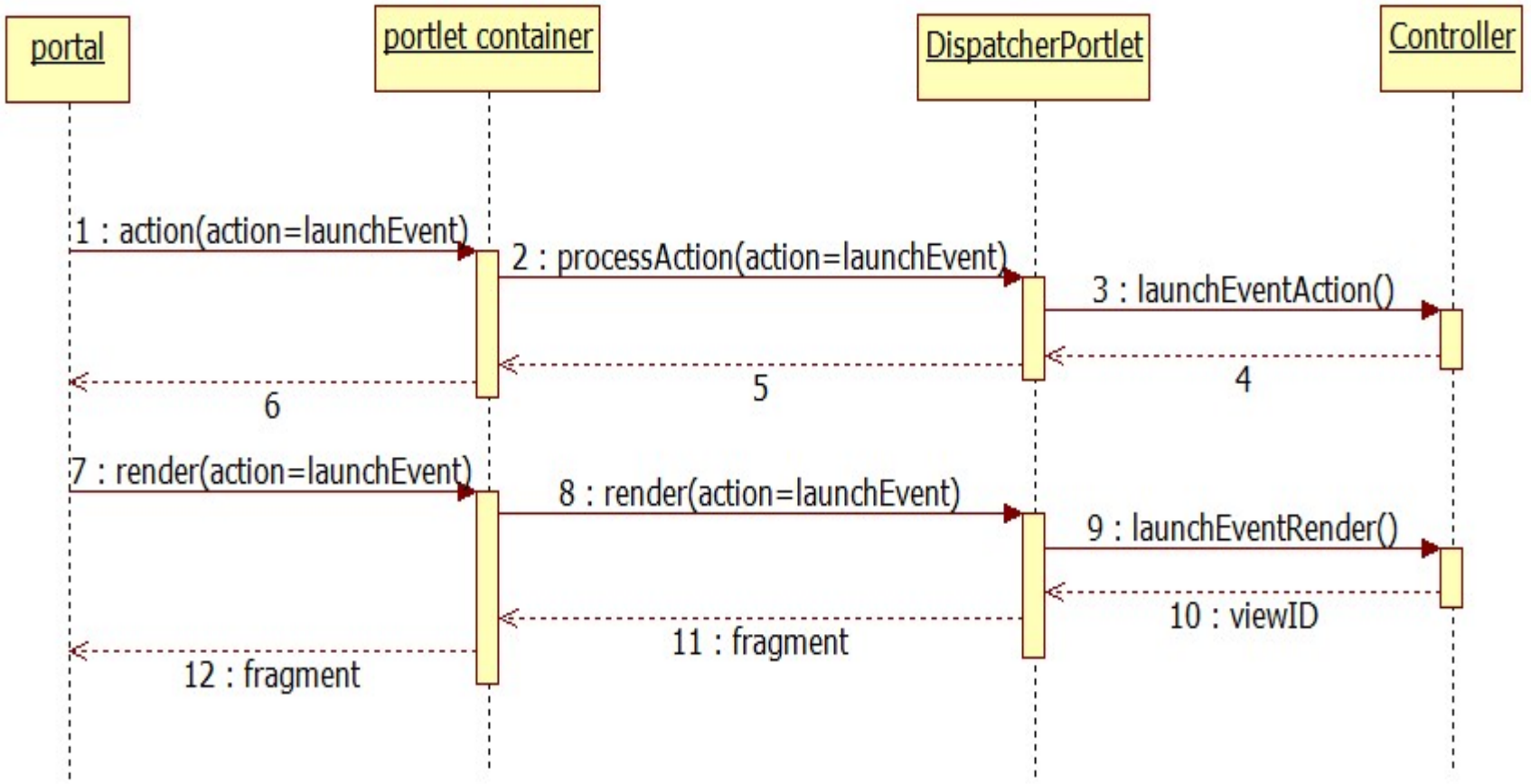
A livello di metodo, indica il metodo del controller da richiamare a partire da una certa action/render request

codice

```
@RequestMapping(params="action=launchEvent")
   public void launchEventAction(ActionRequest request,ActionResponse
response){}

@RequestMapping(params="action=launchEvent")
   public String launchEventRender(RenderRequest
request,RenderResponse response){
      return "index";
   }
```

@RequestMapping - 3



@SessionAttributes

A livello di classe, identifica gli attributi del model da inserire in sessione

codice

```
@SessionAttributes({"search","output"})
public class TestAPortletController {

@RequestMapping
   public String begin(Model model,PortletSession session){
      SearchUserForm search=new SearchUserForm();
     model.addAttribute("search", search);…
```

Il framework colloca automaticamente search in sessione

@ModelAttribute

A livello di parametro, indica che il parametro di un determinato metodo va preso dal model

codice

```
public void jsonChangePage(@ModelAttribute("search")
SearchUserForm search,Model model…)
```

Il parametro search viene preso dal model

@EventMapping

A livello di metodo, identifica il metodo che deve trattare un particolare evento

codice

```
@EventMapping("{http://com.alex/test/portlets/ns}event")
   public void handleEvent(Event event,EventRequest request)
throws Exception {
      …
   }
```

@ResourceMapping

JSR 286

A livello di metodo, identifica il metodo che deve restituire una determinata resource

codice

```
@ResourceMapping(value="changePage")
   public void jsonChangePage(…,
```

```
<portlet:resourceURL var="changePage"
id="changePage">
  <portlet:param name="par1" value="val1"/>
</portlet:resourceURL>
```

Agenda

1. JSR168
2. JSR286
3. Spring MVC Portlet
4. Esempio

Esempio

- IDE: Netbeans 6.9.1

- Spring v.3.0.5

- Maven

- Portlet Container: Apache Pluto

- JQuery