Laboratorio di Basi di Dati e Web

Docente: Alberto Belussi

Lezione 11

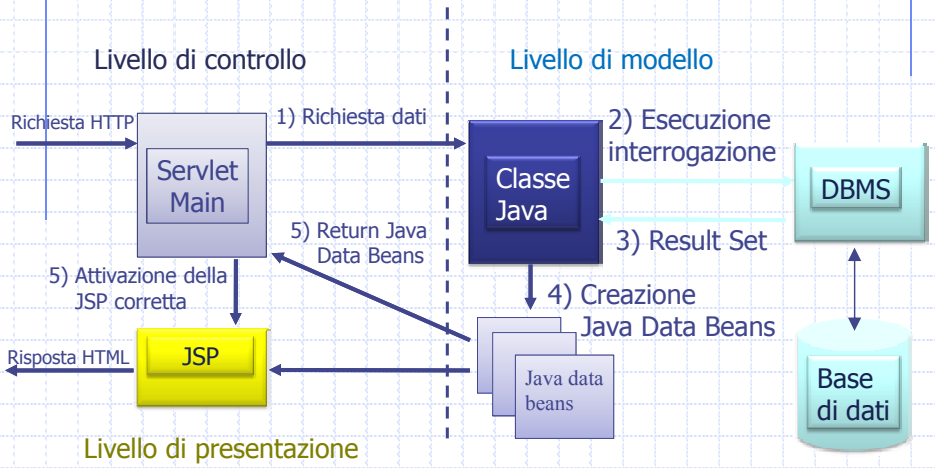
Applicazioni web: evoluzioni

- MVC-2 servlet centric: possibili varianti all'architettura proposta (l'interfaccia **command**).
- Interazione con il DBMS: l'impatto delle **interrogazioni SQL** sulle prestazioni dell'applicazione web
  - Valutazione delle prestazioni di un'interrogazione in postgresql.
  - Gestione indici
- Cenni alle **portlet**: evoluzione della tecnologia servlet/JSP.

Architettura Model-View-Controller (MVC)

- Adottando l'architettura MVC-2 servlet-centric, un'applicazione web risulta composta dai seguenti moduli:
  - Una classe JAVA per l'interazione con il DBMS
  - Un certo numero di bean per gestire il risultato delle interrogazioni
  - Una JSP per ogni schema di pagina
  - Una sola servlet per gestire il flusso di esecuzione.

Approccio Servlet-centric



Variante servlet MAIN

Dividere la servlet main in più classi JAVA che implementano l'interfaccia **command**.

Tale interfaccia include il metodo **execute** che ha la seguente segnatura:

```
public void execute(HttpServletRequest
req) throws CommandException;

public void init(…) throws
CommandException;
```

La main deve solo decidere a quale classe command passare il controllo attraverso l'invocazione del metodo execute. Questo può essere fatto in base al valore di un parametro della richiesta HTTP.

Variante servlet MAIN

Esempio

```
public class Main extends HttpServlet {
        private HashMap commands;
        private DBMS db;
        private static String entToken;

public void init(final ServletConfig config) throws ServletException {
        …
        try {       db = new DBMS();
        } catch (final WebStorageException e) {
        throw new ServletException( "Non è possibile avere una connessione
                al database: " + e.getMessage() );
        }
        Vector classiCommand;
        try {       classiCommand = db.getClassiCommand (…);
        } catch (final WebStorageException e) {
        throw new ServletException( "Main: db.getClassiCommandFac(): " +
                e.getMessage() );
        }
```

Variante sulla servlet MAIN

```
VoceMenuBean voceMenu;
Command      classCommand;
commands = new HashMap();
for (int i=0; i < classiCommand.size(); i++) {
        voceMenu = (VoceMenuBean) classiCommand.get(i);
        try {       classCommand = (Command)
                            Class.forName("it.univr.di.uol.command." +
                            voceMenu.getNomeClasse()).newInstance();
                    classCommand.init(voceMenu);
                    commands.put(voceMenu.getNome(), classCommand);
        } catch (final Exception e) {
              final String error = "Errore generico nel caricare la classe " +
                            voceMenu.getNomeClasse() +
                            ": " + e.getMessage();
        throw new ServletException(error);
        }
    }
}
```

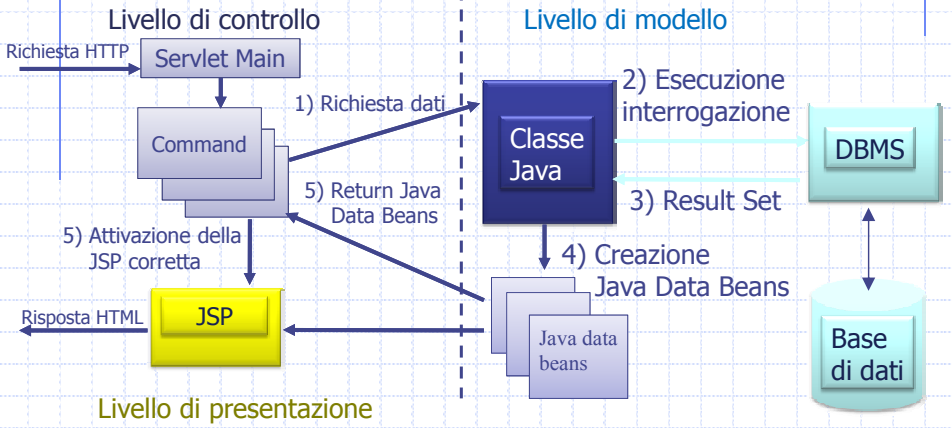
Variante sulla servlet MAIN

```
public void service(final HttpServletRequest req,
                            final HttpServletResponse res)
                            throws ServletException, IOException
{
…
    final Command cmd = commands.get(req.getParameter(entToken));
    cmd.execute(req);

    final RequestDispatcher rd =
                            getServletContext().getRequestDispatcher(
                                    fileJsp+"?"+req.getQueryString() );
    rd.forward(req, res);
}
```

Classi command

```
public class PersonaCommand implements Command {
   static String nomeFileElenco = "/jsp/elencoPersone.jsp";

public void init(VoceMenuBean voceMenu) throws CommandException {

   …
   }

public void execute(HttpServletRequest req) throws CommandException {

   …
   }

}
```

Approccio Servlet-centric with command pattern



Ottimizzazione delle interrogazioni SQL

- Interrogazioni SQL complesse possono rallentare la risposta di una applicazione web.
- Per analizzare ed intervenire sull'esecuzione di una interrogazione SQL, postgresql mette a disposizione il comando EXPLAIN <SQL query>.

Comando EXPLAIN

EXPLAIN [ANALYZE] [VERBOSE] *statement*

Questo comando mostra il piano di esecuzione che il modulo di ottimizzazione di PostgreSQL genera per lo statement generato.

Il piano mostra le tabelle coinvolte nell'interrogazione, gli algoritmi applicati per la scansione e per il join tra tabelle, gli indici usati, ecc…

Comando EXPLAIN

La parte più critica e interessante del piano riguarda la stima dei costi di esecuzione dell'interrogazione.

Tale costo viene misurato in accessi a pagine della memoria secondaria.

In realtà vengono forniti due numeri: il primo rappresenta il costo prima di produrre la prima riga di risultato, il secondo il costo per restituire tutte le righe del risultato.

Comando EXPLAIN

Esempio 1

EXPLAIN SELECT nome,cognome,datanascita FROM persona where nome='Alberto';

```
"Seq Scan on persona  (cost=0.00..114.03 rows=4 width=20)"
"  Filter: ((nome)::text = 'Alberto'::text)"
```

Costruiamo un indice sull'attributo nome:

CREATE INDEX persona_nome ON persona(nome) e rilanciamo il comando EXPLAIN sopra riportato

```
"Bitmap Heap Scan on persona  (cost=4.28..17.46 rows=4 width=20)"
"  Recheck Cond: ((nome)::text = 'Alberto'::text)"
"  ->  Bitmap Index Scan on persona_nome  (cost=0.00..4.28 rows=4 width=0)"
"        Index Cond: ((nome)::text = 'Alberto'::text)"
```

Comando EXPLAIN

Esempio 2

EXPLAIN SELECT * FROM inserogato i JOIN docente d ON i.id=d.id_inserogato;

```
"Hash Join  (cost=3554.44..5970.43 rows=13389 width=624)"
"  Hash Cond: (d.id_inserogato = oi.id)"
"  ->  Seq Scan on docente d  (cost=0.00..279.89 rows=13389 width=117)"
"  ->  Hash  (cost=1789.75..1789.75 rows=22775 width=507)"
"        ->  Seq Scan on inserogato oi  (cost=0.00..1789.75 rows=22775 width=507)"
```

Costruiamo un indice sull'attributo id_inserogato e rilanciamo il comando EXPLAIN sopra riportato

```
"Merge Join  (cost=0.34..4864.54 rows=13389 width=624)"
"  Merge Cond: (oi.id = d.id_inserogato)"
"  ->  Index Scan using occorrenzains_pkey on inserogato oi
                                        (cost=0.00..3797.54 rows=22775 width=507)"
"  ->  Index Scan using docente_id0_occorrenzains_index on docente d
                                        (cost=0.00..843.05 rows=13389 width=117)"
```

Comando EXPLAIN

Si noti che il comando EXPLAIN non è SQL standard. Lo troviamo solo su postgresql.

E' possibile eseguire un EXPLAIN anche dall'interfaccia grafica di pgAdmin.

Qualndo si aggiunge l'opzione ANALYSE allora l'interrogazione viene eseguita effettivamente e viene restituito il tempo effettivo di esecuzione.

Portlet

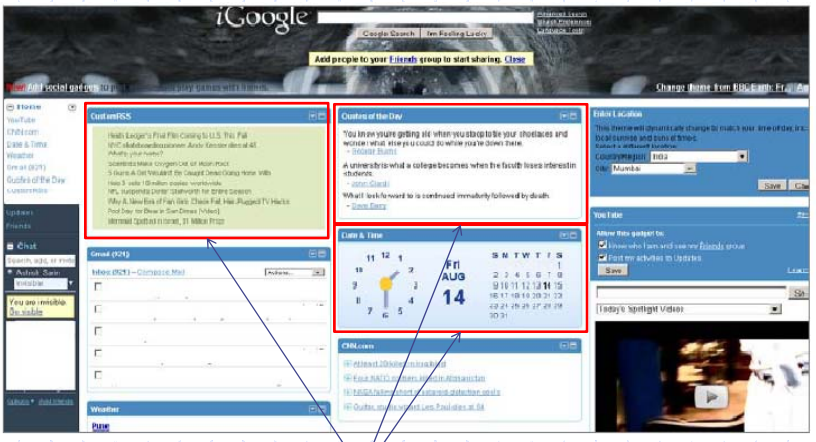
Java Portlet technology provides a standard approach to incorporating **user experience features** in your web application, which includes **consistent look and feel**, **personalization**, **customization** and **content aggregation.**

Portlet and Portal

A portal is a collection of mini web applications, called *portlets,* which supports features like personalization, content aggregation, authentication and customization.

Portlets act as windowed web applications within the portal and each window on a portal web page (called portal page) represents a portlet.

Portal example



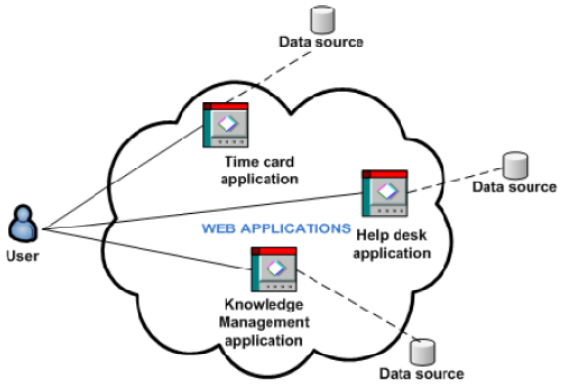
Portal benefits

Enriched User Experience

Developing a web portal makes a good business case if it's required to gather and present information from various data sources, applications and systems, to give a unified view of the information to the user based on his identity.

Portal benefits

Enriched User Experience Example



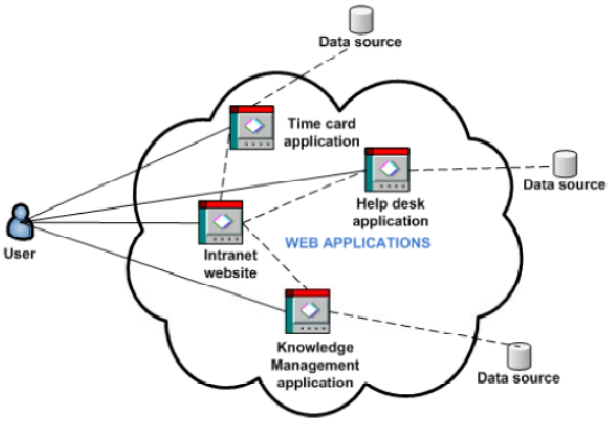
Portal benefits

Let' s say, the organization goes one step ahead and provides a **single sign-on solution** and access to the different web applications from an intranet website.

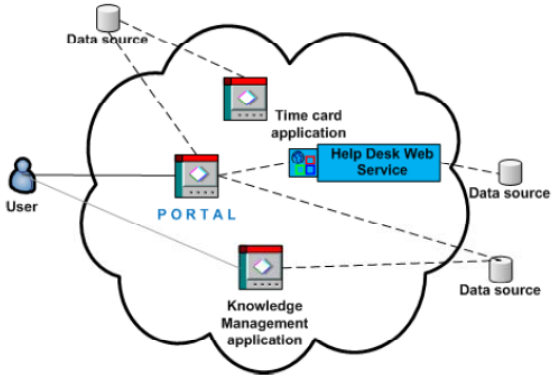
By providing single sign-on feature, the organization has provided easy access to the web applications, but you still need to filter the information that interests you.

Portal benefits

Enriched User Experience Example



Portal benefits



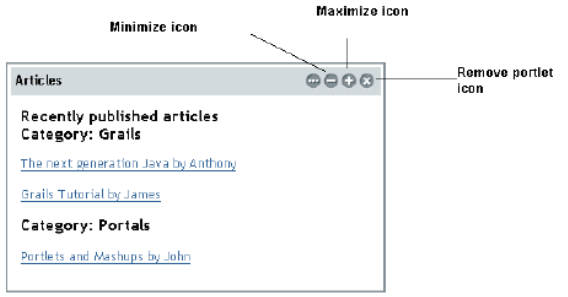
Usually, portals provide the most-used features of the original web application to the user and when least-used features of the web application are requested then the portal redirects the user to the original web application for performing such action(s).

Portlet (definition)

A **portlet** is a pluggable user interface component which provides specific piece of content, which could be a service or information from existing information systems.

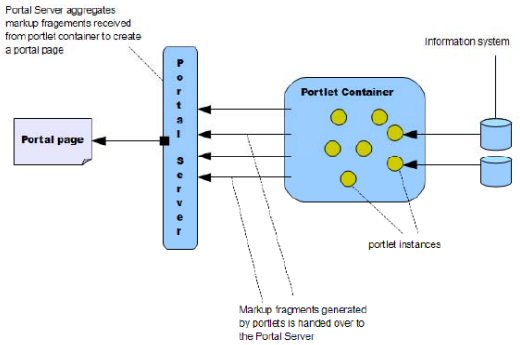
Portlet components are responsible for providing the user interface of the portal by **accessing distinct applications, systems or data sources** and **generating markup fragment to present the content to the portal users**.

Portlet



Portal infrastructure

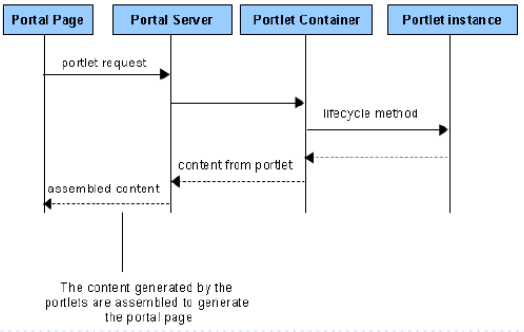
A portlet on a portal page is represented by a portlet instance inside the portlet container.



Portal infrastructure

A portal server is responsible for submitting user requests received from the portal page to the portlet container and aggregating response generated by portlets to form the portal page.

Therefore, the responsibility of providing consistent look and feel for the portal lies with the portal server.

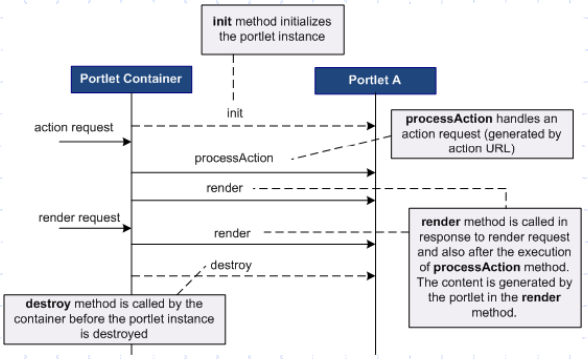


Portlet lifecycle

Portlet lifecycle methods defined in the Portlet interface.

An action request results in invocation of processAction method followed by render method.

A render request results in invocation of render method. The timing of init and destroy methods invocation is dependent on portlet container implementation.



Portlet interface

INIT METHOD

The **init** method is invoked by the portlet container after the portlet is loaded and instantiated by the portlet container.

The method gives an opportunity to the portlet instance to initialize itself before processing any request from the client.

The signature of the init method is:

```
void init(PortletConfig config) throws PortletException
```

Portlet interface

RENDER METHOD

The **render** method is invoked by the portlet container when a render request is received for the portlet instance. The render method is responsible for generating content that forms part of a portal page. The signature of the init method is:

```
void render(RenderRequest request, RenderResponse response)
                              throws PortletException, IOException
```

Portlet interface

PROCESSACTION METHOD

The processAction method is invoked in response to an action request. The processAction method represents a user action which results in state change, like submitting an order request form.

The signature of the processAction method is:

```
void processAction( ActionRequest request,
                    ActionResponse response) throws
                          PortletException, IOException
```

Portlet interface

DESTROY METHOD

The destroy method is invoked by the portlet container before removing the portlet instance from memory. The destroy method is the cleanup method in portlets where the instance may release any held resources (like database connections, EJB references) or save its state to a persistent storage (like database or file).

The signature of destroy method is:

```
void destroy()
```

Riferimenti

- Ashish Sarin. "PORTLET in action". Mannin, 2010 (in uscita).