CORSO DI LAUREA IN INFORMATICA

TECNOLOGIE SOFTWARE PER IL WEB

SERVLET – III PARTE

a.a 2022/23

Servlet context

https://jakarta.ee/specifications/servlet/6.0/apidocs/jakarta.servlet/jakarta/servlet/servletcontext

- Ogni Web application esegue in un **contesto** :
  - **C'è corrispondenza 1:1 tra una web application e suo contesto** (ovvero una memoria "globale" all'interno dell'applicazione che contiene informazioni sull'applicazione stessa ed eventuali parametri da tenere gloabli)
- Una qualsiasi servlet di una web application può accedere al contesto dell'applicazione tramite l'interfaccia **ServletContext.**
- Si può ottenere un'istanza di tipo ServletContext all'interno della Servlet utilizzando il metodo **getServletContext()**
- *IMPORTANTE:* la memoria usata dal servlet context è **condivisa** tra **tutti** gli **utenti,** le **richieste** e le **servlet** facenti parte della stessa Web application

Risalire al path

getServletContext().getContextPath()

Restituisce la cartella principale dell'applicazione:
/MostraRequestHeaders-1.0-SNAPSHOT

getServletContext().getRealPath("/index.html")

Restituisce il path reale della risorsa "/index.html":
/Users/marello/apache-tomcat-10.1.6/webapps/MostraRequestHeaders-1.0-SNAPSHOT/index.html

Parametri di inizializzazione del contesto

- Parametri di inizializzazione del contesto definiti all'interno di elementi di tipo context-param in web.xml

```
<web-app>
  <context-param>
     <param-name>feedback</param-name>
     <param-value>feedback@deis.unibo.it</param-value>
  </context-param>
  ...
</ web-app >
```

- Sono accessibili a tutte le Servlet della Web application

```
…
ServletContext ctx = getServletContext();
String feedback =
ctx.getInitParameter("feedback");
…
```

Attributi di contesto

- Gli attributi di contesto sono accessibili a tutte le servlet e funzionano come variabili "*globali*"
- Vengono gestiti a runtime:
  - possono essere creati, scritti e letti dalle Servlet
- Possono contenere oggetti anche complessi (serializzazione/deserializzazione)

scrittura

```
ServletContext ctx = getServletContext();
ctx.setAttribute("utente1", new User("Giorgio Bianchi"));
ctx.setAttribute("utente2", new User("Paolo Rossi"));
```

lettura

```
ServletContext ctx = getServletContext();
Enumeration aNames = ctx.getAttributeNames();
while (aNames.hasMoreElements)
{
  String aName = (String)aNames.nextElement();
  User user = (User) ctx.getAttribute(aName);
  ctx.removeAttribute(aName);
}
```

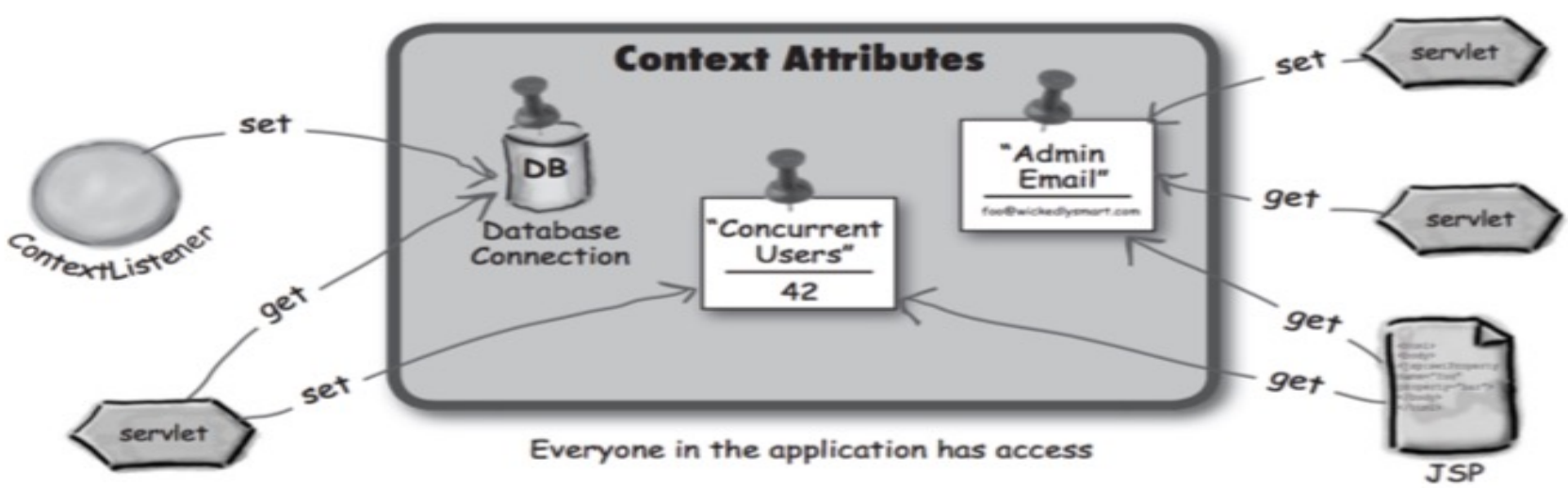
Attenzione: scope DIFFERENZIATI (scoped objects)

- Gli oggetti di tipo ServletContext, HttpSession, HttpServletRequest forniscono metodi per immagazzinare e ritrovare oggetti nei loro rispettivi ambiti (*scope*)
- Lo scope è definito dal **tempo di vita** (lifespan) e dall'**accessibilità** da parte delle Servlet

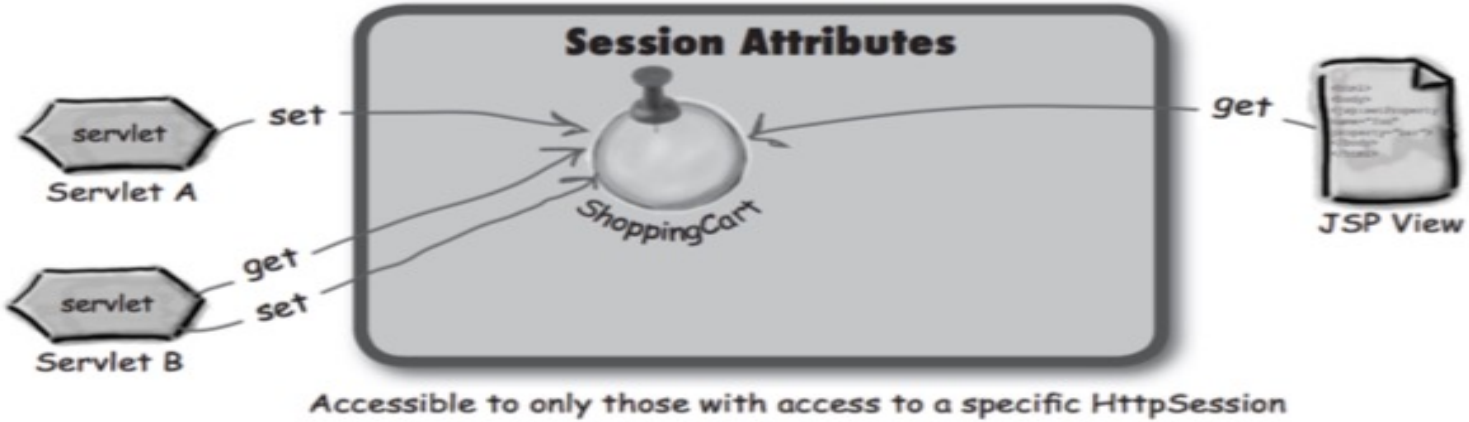
Ambito	Interfaccia	Tempo di vita	Accessibilità
Request	`HttpServletRequest`	**Fino all'invio della risposta**	**Servlet corrente e ogni altra pagina inclusa o in forward**
Session	`HttpSession`	**Lo stesso della sessione utente**	**Ogni richiesta dello stesso client**
Application	`ServletContext`	**Lo stesso dell'applicazione**	**Ogni richiesta alla stessa Web app anche da clienti diversi e per servlet diverse**

Funzionalità degli scoped object

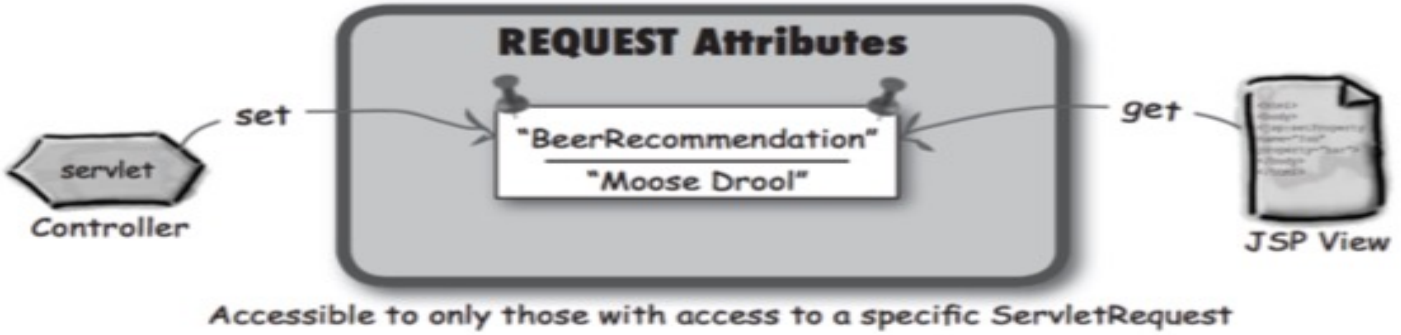
- Gli oggetti con scope forniscono i seguenti metodi per immagazzinare e ritrovare oggetti nei rispettivi ambiti (scope):
  - **void setAttribute(String name, Object o);**
  - **Object getAttribute(String name);**
  - **void removeAttribute(String name);**
  - **Enumeration getAttributeNames();**



Tutti gli utenti
E quante volte necessario



Solo l'utente di quella sessione
E quante volte necessario



Solo l'utente che ha inviato la richiesta
Ed una sola vlta

Ridirezione del browser

- È possibile inviare al browser una risposta che lo forza ad accedere ad un'altra pagina/risorsa (*ridirezione*)
- Possiamo ottenere agendo sull'oggetto **response** invocando il metodo
  - **public void sendRedirect(String url);**
- *Scenario: In un sito di commercio elettronico, una volta che selezioniamo il prodotto, siamo pronti per l'acquisto e clicchiamo su "paga", il browser reindirizza alla rispettiva pagina di pagamento online. Qui, la risposta del sito di shopping lo reindirizza alla pagina di pagamento e un nuovo URL può essere visto nel browser.*
- Es:

```
String name=request.getParameter("name");
response.sendRedirect("https://www.google.co.in/#q="+name);
```

Scrittura della risposta da parte di più servlet o JSP

La jsp o servlet chiamata tramite forward creerà la risposta e la invierà al servlet container (chiudendo di fatto l'esecuzione dell'applicazione)
Al di fuori di MVC, una servlet potrebbe voler scrivere essa stessa una parte della risposta e «farsi aiutare» da una jsp o altra servlet a scrivere il resto (sempre nello stesso oggetto risposta). Quindi, dopo aver scritto qualcosa nella risposta, può passare il controllo ad un'altra servlet. In questo caso il controllo viene passato sempre tramite dispatcher ma si usa invece il metodo include (in questo caso è VIETATISSIMO usare forward)
La jsp o servlet chiamata tramite include aggiungerà il suo output alla risposta ed alla sua fine esecuzione restituirà automaticamente il controllo alla servlet che l'ha chiamata. (L'oggetto risposta è unico e verrà condiviso)

Includere una risorsa (include)

- Per includere una risorsa si ricorre a un oggetto di tipo **RequestDispatcher** che può essere richiesto al contesto indicando la risorsa da includere.
- Si invoca quindi il metodo **include** passando come parametri **request** e **response** che vengono così condivisi con la risorsa inclusa

può essere anche una pagina JSP

```
RequestDispatcher dispatcher =
  getServletContext().getRequestDispatcher("/inServlet");
dispatcher.include(request, response);
```

RequestDispatcher pluto = request.getRequestDispatcher("/WEB-INF/results/showHeaders.jsp");
pluto.include(request, response);

Inoltro (forward)

- Si usa in situazioni in cui una Servlet si occupa di parte dell'elaborazione della richiesta e delega a qualcun altro la gestione della risposta
  - *Attenzione:in questo caso la **risposta** è di **competenza esclusiva** della risorsa che riceve l'inoltro*
  - Se nella prima Servlet è stato fatto un accesso a ServletOutputStream o PrintWriter si ottiene una IllegalStateException (ovvero, se anche la prima Servlet vuole scrivere nella risposta)
- Si deve ottenere un oggetto di tipo **RequestDispatcher** da request passando come parametro il nome della risorsa
- Si invoca quindi il metodo **forward** passando anche in questo caso **request** e **response**

può essere anche una pagina JS

```
RequestDispatcher dispatcher =
  getServletContext().getRequestDispatcher("/inServlet");
dispatcher.forward(request, response);
```

Include e forward