SUNBEAM

Java EE™

Advanced Java

Trainer: Nilesh Ghule

servlet init params

① servlet specific settings/configuration.

② given in web.xml
<init-param> or @WebServlet

③ Loaded in ServletConfig obj associated with each Servlet object.

val = config.getInitParameter("key")

④ string values.

⑤ until Servlet is destroyed.

request params

① send data with request from client to server.

② given in req body
→ submit form method=post

OR

given in url
→ submit form method=get
or query string

③ accessed using req obj:

val = req.getParameter("—");
vals = req.getParameterValues("—");

④ string values.

⑤ until request is completed i.e. response is generated.

request attributes

① send data from one servlet to another while forwarding/including req (RequestDispatcher)

② first Servlet:

req.setAttribute("key", value);
↓
Object

③ accessed in next servlet:

value = req.getAttribute("key");
↓
Object

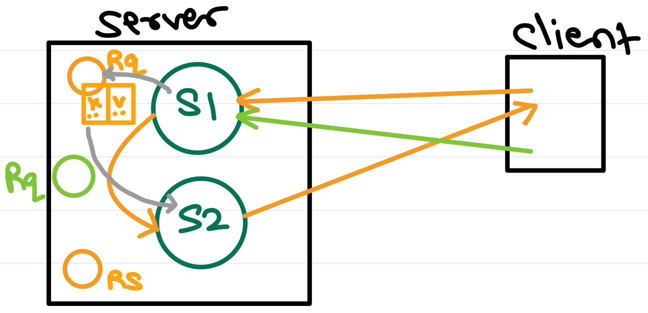
④ Object values

⑤ until request is completed i.e. response is generated.

Scopes

Request

```
req.setAttribute("key",value);
value = req.getAttribute("key");
```

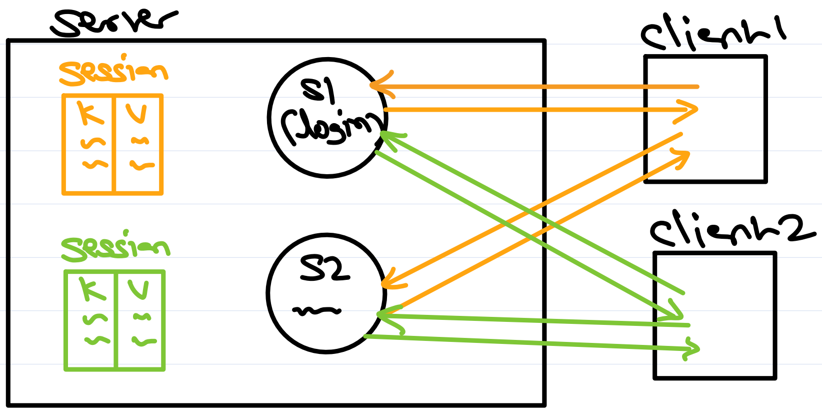


req attributes are accessible only for current req i.e. req getting forwarded/included to next component. When resp is generated, req & resp objects are destroyed.

Session

A session is created for each user.

```
session.setAttribute("key", value);
value = session.getAttribute("key");
```

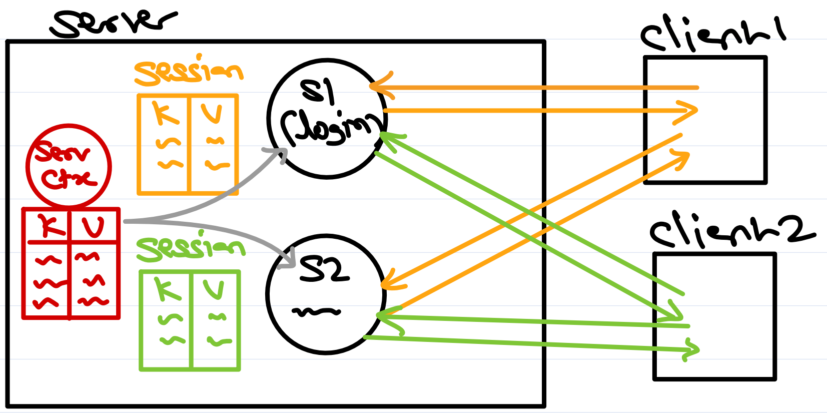


session attributes are accessible in current user session i.e. available for all reqs to all pages by the same user.

Application

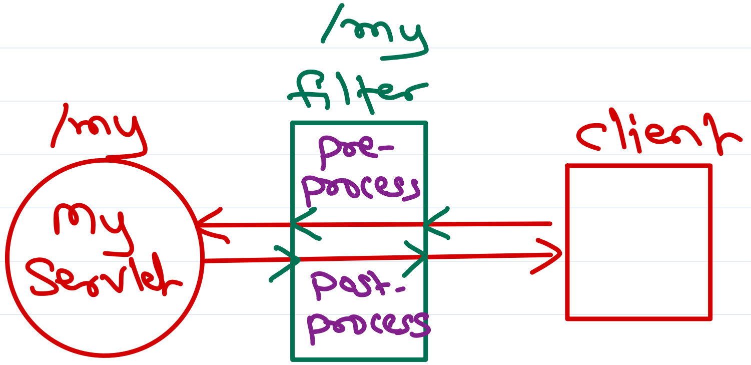
A ServletContext is created for each web appln.

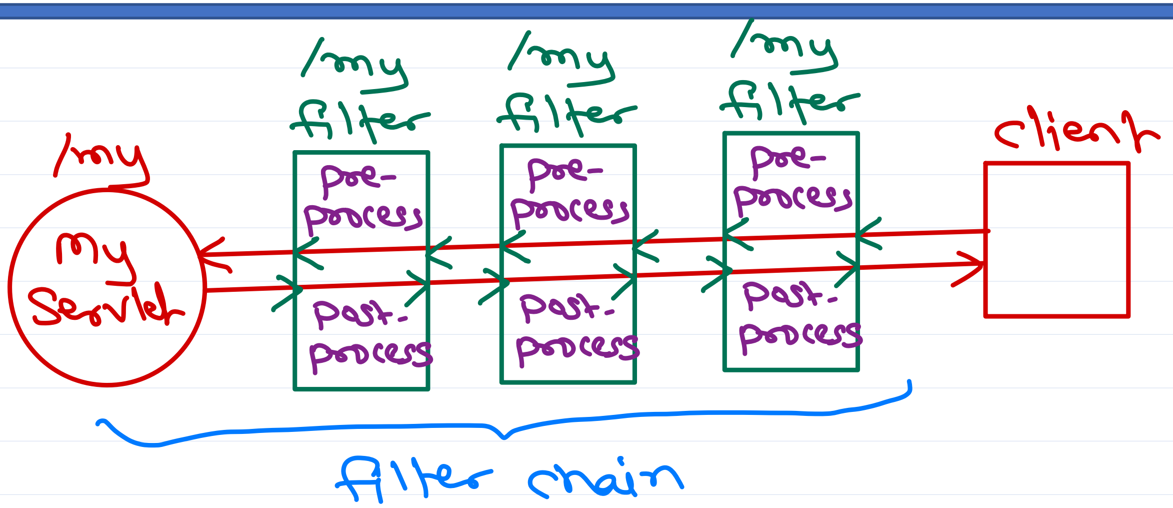
```
ctx.setAttribute("key", value);
value = ctx.getAttribute("key");
```



Servlet ctx attributes are accessible throughout the appln (like global variables) i.e. to all reqs to all pages by all users.

Filters



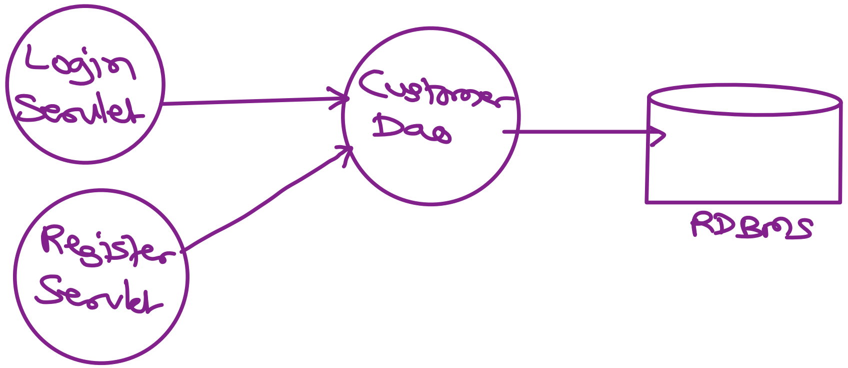


```
@WebFilter("/my")
class MyFilter implents Filter {
    void init(FilterConfig cfg) ... {
        ~ //one-time init
    }
    void destroy() {
        ~ // one-time deinit
    }
    void doFilter(req, resp, chain) ... {
        //pre-processing
        chain.doFilter(req, resp);
        // post-processing
    }
}
```

Filters

AOP
Aspect Oriented Programming

- Filters is way of implementing AOP in Java EE applications. Filters are used to perform pre-processing, post-processing or both for each request.
- Multiple filters can be executed in a chain/stack before/after handling request.
- javax.servlet.Filter interface is used to implement Filters.
  - void init(FilterConfig filterConfig);
  - void doFilter(ServletRequest req, ServletResponse resp, FilterChain chain);
  - void destroy();
- Can be configured with @WebFilter or in web.xml (similar to servlets).



✓ AOP is implementation of cross cutting concerns without modifying business logic.

✓ cross-cutting concerns = extra functionality

- logging → pre or post
- monitoring → post
- profiling → pre and post
- tx mgmt ← pre and post
- time - post = diff
- security → pre

success → commit()

exception → rollback

pre → autoCommit(false)

✓ CCC impl → Aspect → Advice

Aspect → class

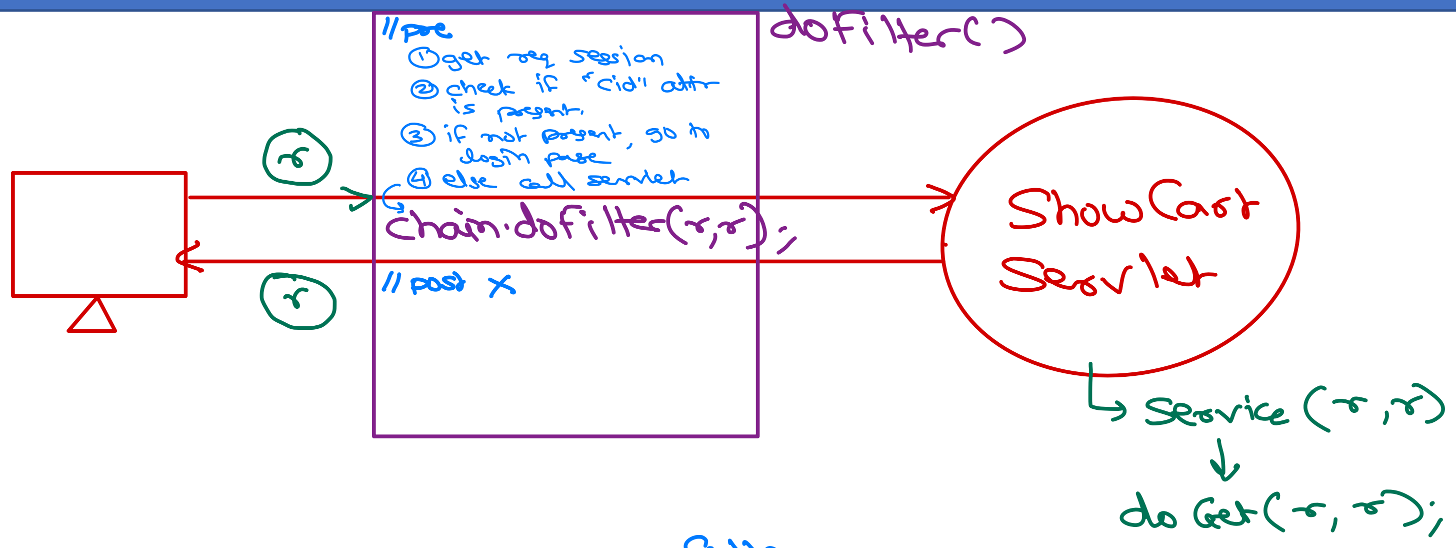
Advice → method

- → pre-processing
- → post-processing
- → pre & post

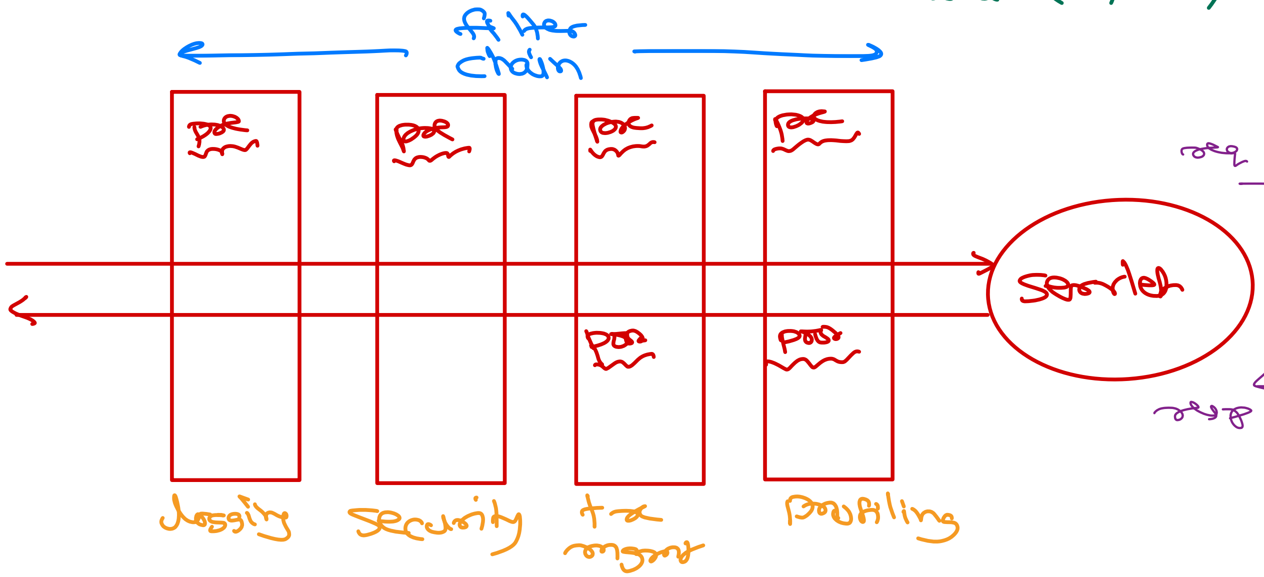
(Java) AOP
- → Java proxy
- → Aspect/J
- → Spring AOP
- → Java EE filters

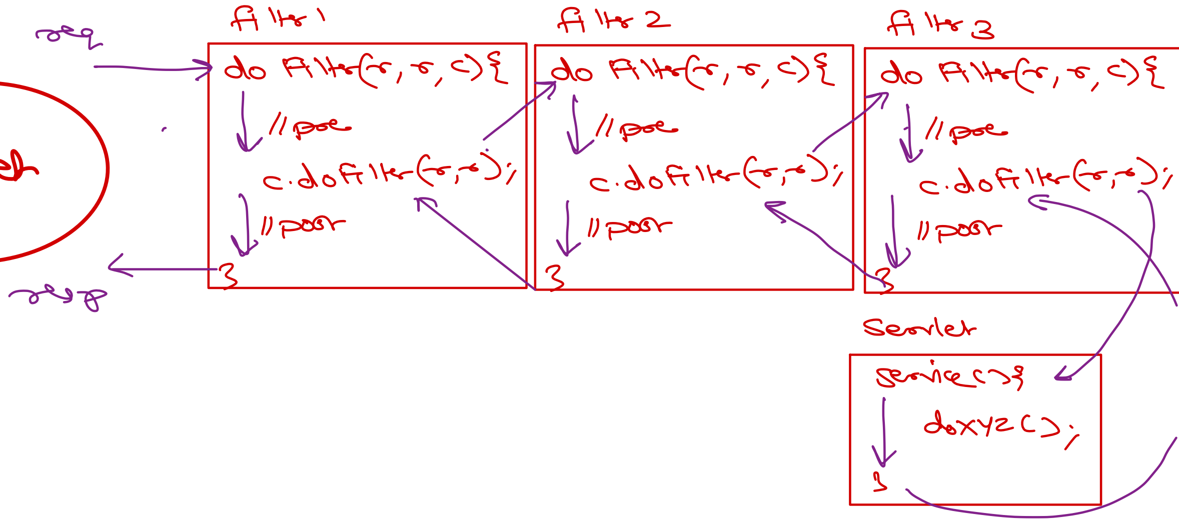
url-pattern = /*

Security Filter



```
//pre
 ① get req session
 ② check if "cid" attr
    is present.
 ③ if not present, go to
    login page
 ④ else call servlet
chain.dofilter(r,r);
//post X
```





```
filter 1
do Filter(r, r, c){
  //pre
  c.dofilter(r,r);
  //post
}

filter 2
do Filter(r, r, c){
  //pre
  c.dofilter(r,r);
  //post
}

filter 3
do Filter(r, r, c){
  //pre
  c.dofilter(r,r);
  //post
}

Servlet
service(){
  doxyz();
}
```

Listeners

server-side

- Listeners are used to handle application level events.
- There are many listener interfaces.
  - ✓ ServletContextListener ✗
  - ✓ HttpSessionListener ✗
  - ✓ ServletRequestListener
  - ✓ ServletContextAttributeListener
  - ✓ HttpSessionActivationListener
  - ✓ HttpSessionAttributeListener
  - ✓ ServletRequestAttributeListener
- Listener class must implement one or more listener interface.
- Can be configured with @WebListener or in web.xml.

request begin }
request end

Context initialized }
Context destroyed

Session Created }
Session destroyed

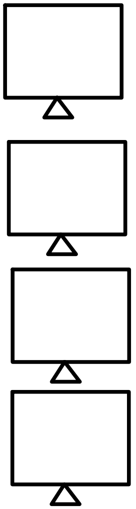
attrib added }
attrib removed
attrib modified

```
Context Initialized:                Integer
   ctx.setAttr("cnt", 0);

session Created:
   cnt = ctx.getAttr("cnt");
   cnt++;
   ctx.setAttr("cnt", cnt);

session Destroy:
   cnt--;
```

Assign 4



keep track of num of online users at any moment.

```
<listener>
  <listener-class>pkg.MyListener</listener-class>
</listener>
```

SUNBEAM

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>