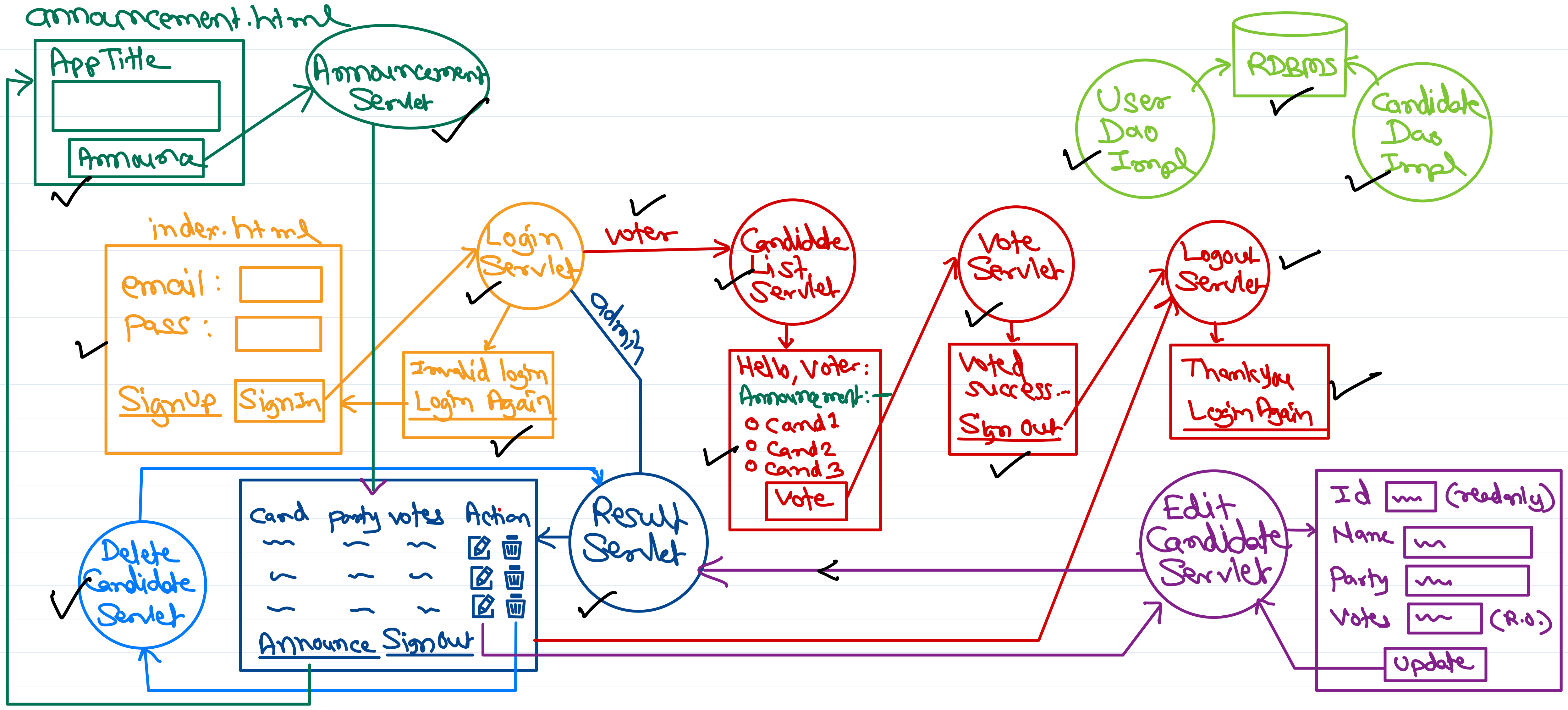
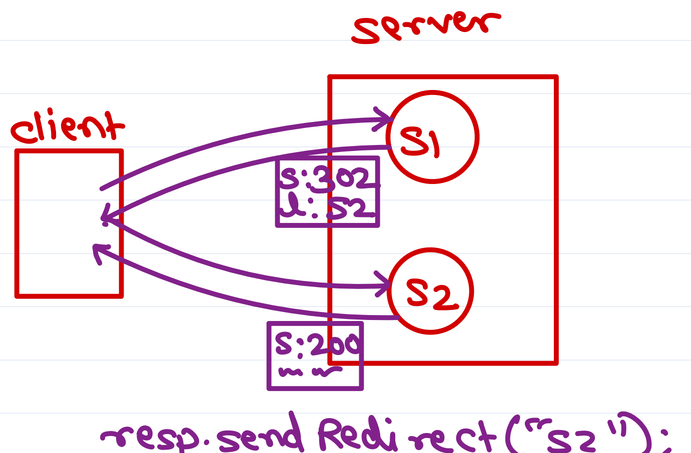
Advanced Java

Trainer: Nilesh Ghule

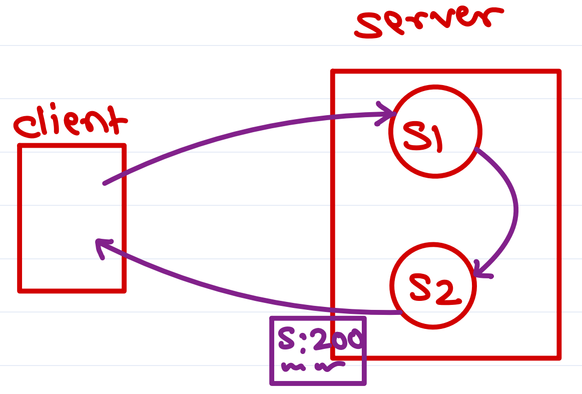
Election Management



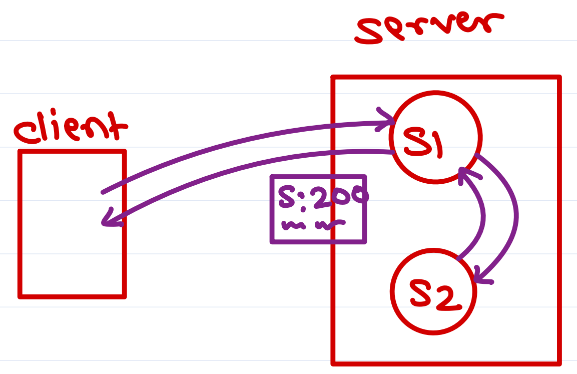
Inter-Servlet Communication



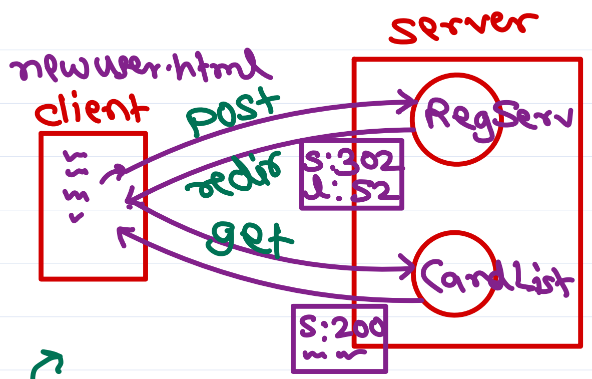
resp.sendRedirect("s2");



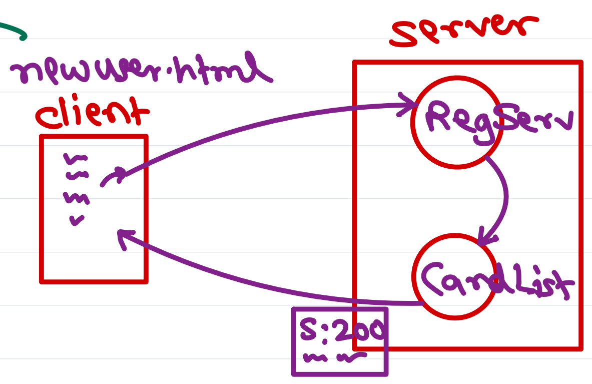
rd = req.getRequestDispatcher("s2");
rd.forward(req, resp);



rd = req.getRequestDispatcher("s2");
rd.include(req, resp);



when client refresh, last req (i.e. post) is replayed, due to which data is sent twice & may be added or updated in db twice.



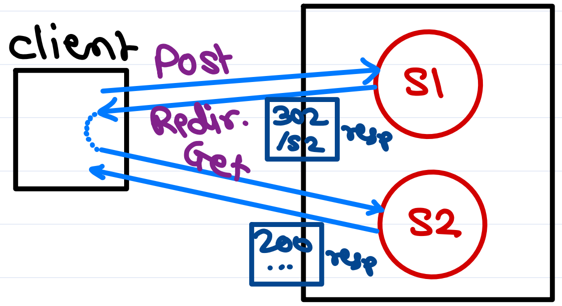
PRG pattern.

if client refresh, only last req (GET) is replayed & data fetched again. Earlier data not posted twice.

Servlet Communication/Navigation

Http Redirection

resp.sendRedirect("url");

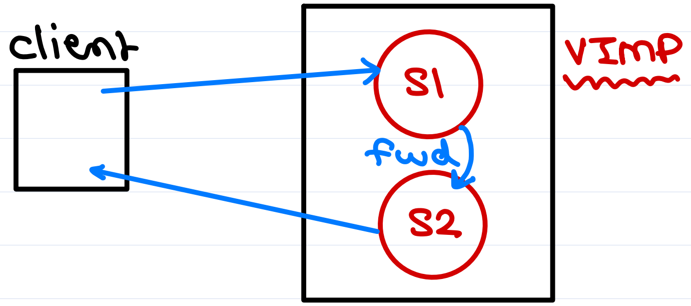


① execution as shown in diag. Two diff req from browser - slower.
② url in browser is changed.
③ can navigate to any page in or outside the web appln.
④ Useful after Post req, so that same data is not posted again upon refresh.
Redirection → PRG pattern.

forward() ← RequestDispatcher → include()

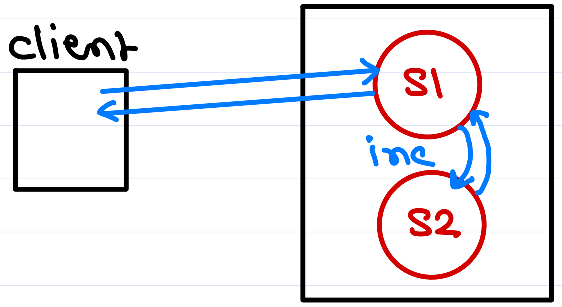
rd = req.getRequestDispatcher("url");

rd.forward(req, resp);



① same req is fwded to next page - faster.
② url in browser is unchanged - browser not aware of navigation.
③ can navigate to any page inside the web appln.
④ final resp generated by next page.

rd.include(req, resp);



① same req is fwded to next page - faster.
② url in browser is unchanged - browser not aware of navigation.
③ can navigate to any page inside the web appln.
④ final resp is sent by first page.

State Management

* maintaining state/info about client is called as "state mgmt".

* Client side state mgmt

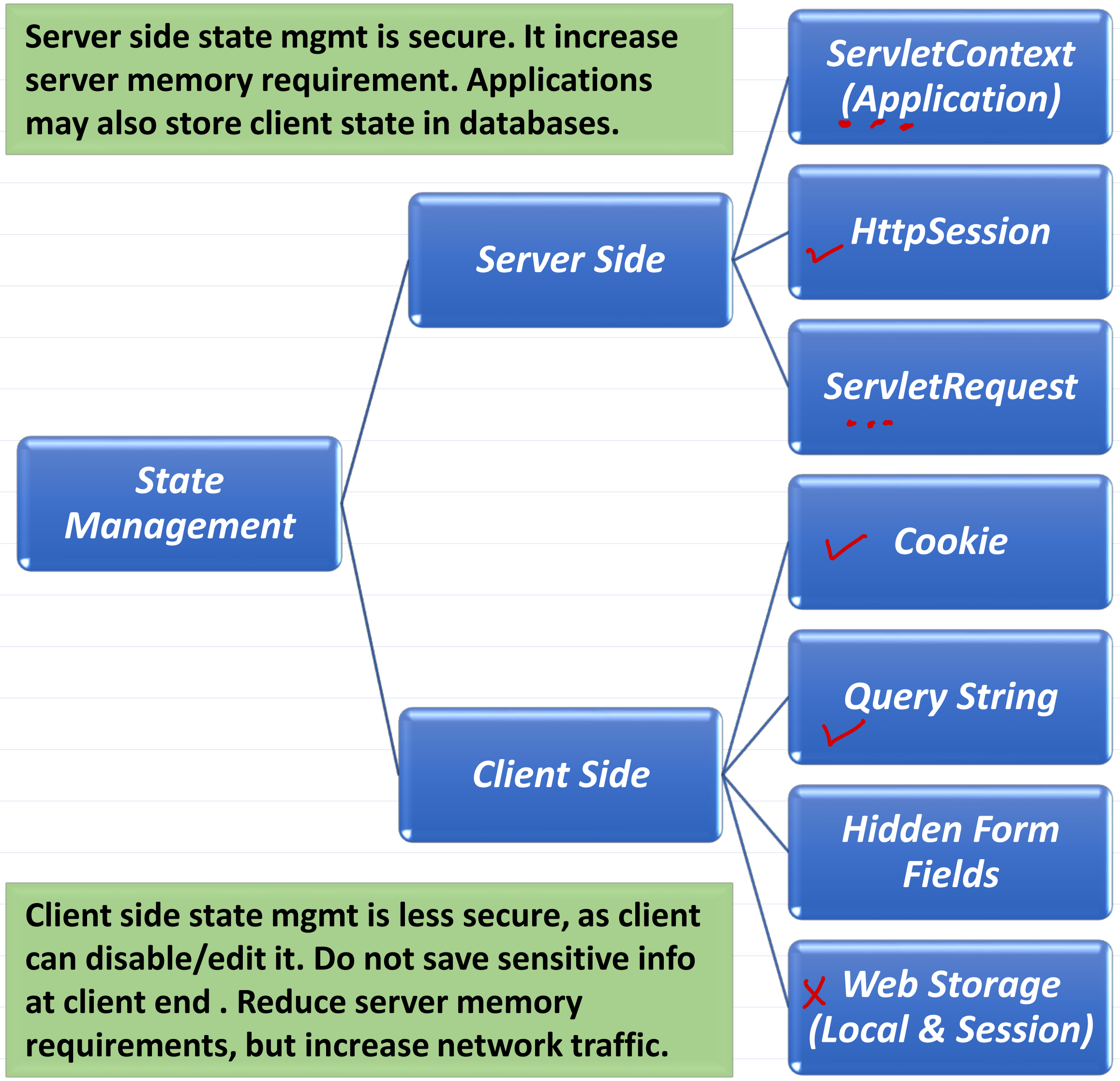
- Cookie	✓ Need less server resources.
- QueryString	× Visible to client
- Hidden fields	× Client can tamper
~~- HTML Storage~~	

* Server side state mgmt

- Session	✓ Secure (Not accessible to Client)
- Request	× Need more server resources.
- Application a.k.a. ServletContext	

Java Web Applications – State management – at a glance

Server side state mgmt is secure. It increase server memory requirement. Applications may also store client state in databases.



Client side state mgmt is less secure, as client can disable/edit it. Do not save sensitive info at client end . Reduce server memory requirements, but increase network traffic.

ServletContext is created one for the application. It stores all meta-information like context param and attributes (set of key-value pairs). Accessible in all web components for all requests of all clients.

HttpSession is created one for ~~the application~~ each client. It stores attributes (key-value pairs) for each client. Accessible in all web components for all requests from the same client. Session tracking is done using cookie or URL rewriting or SSL.

ServletRequest is created one for ~~the application~~ each request. It stores headers and attributes (key-value pairs) for the request. Accessible in all web component to which same request is forwarded/included (using RequestDispatcher).

Cookie is text data (key-value pair) sent from application to client. Client (browser) send it back to the server with each request. Cookies stored in browser memory and disappear when browser is closed. Persistent cookies are stored on client disk.

Query string is key-value pair appended to the URL like url?key=value. Commonly used in master detail pages to load details of selected items from the master.

Data can be stored in HTML <input type="hidden" ...>. It will be sent back to server when HTML form is submitted. Typically used to keep identity of currently selected/displayed item.

Application stores key-value data in client browser. LocalStorage has no expiration date, while SessionStorage expires when browser is closed. Both are managed in client-side JS code using window.localStorage & window.sessionStorage.

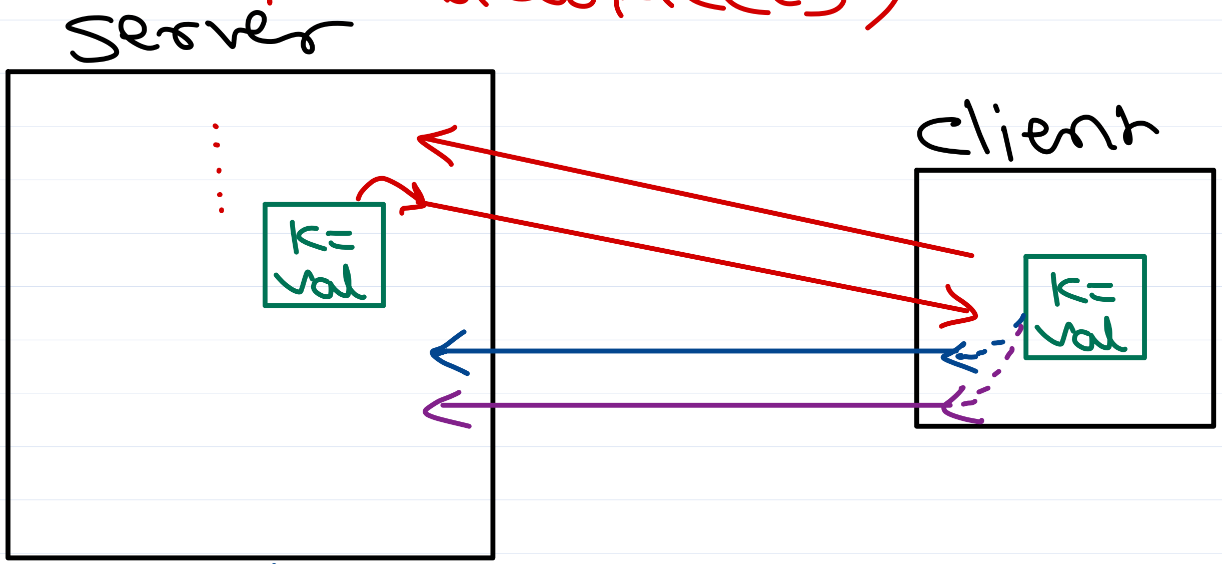
Cookie

Cookie → text key value pair
- stored on client side
    - temp cookie - browser memory.
    - persistent cookie - client disk until expiry time.
- max size = 4 KB
- created by server and sent to client in a response.
- afterwards cookie is sent back to server with each request by the client.
- cookies can be seen and modified by the client.
- Cookies can be disabled in browser.

c.setMaxAge(-1); → Cookie become temp.
c.setMaxAge(0); → cookie delete

* Send cookie to client:

```
Cookie c = new Cookie("k", "v");
c.setMaxAge(seconds); //← persistent cookie
resp.addCookie(c);
```



* receive cookie from client:

```
Cookie[] arr = req.getCookies();
for(Cookie c : arr) {
    if(c.getName().equals("k")){
        v = c.getValue();
        break;
    }
}
```

Session

HttpSession is key-value map stored on server side for each user.

To create/access session:

session = req.getSession();

To store key-value in session:

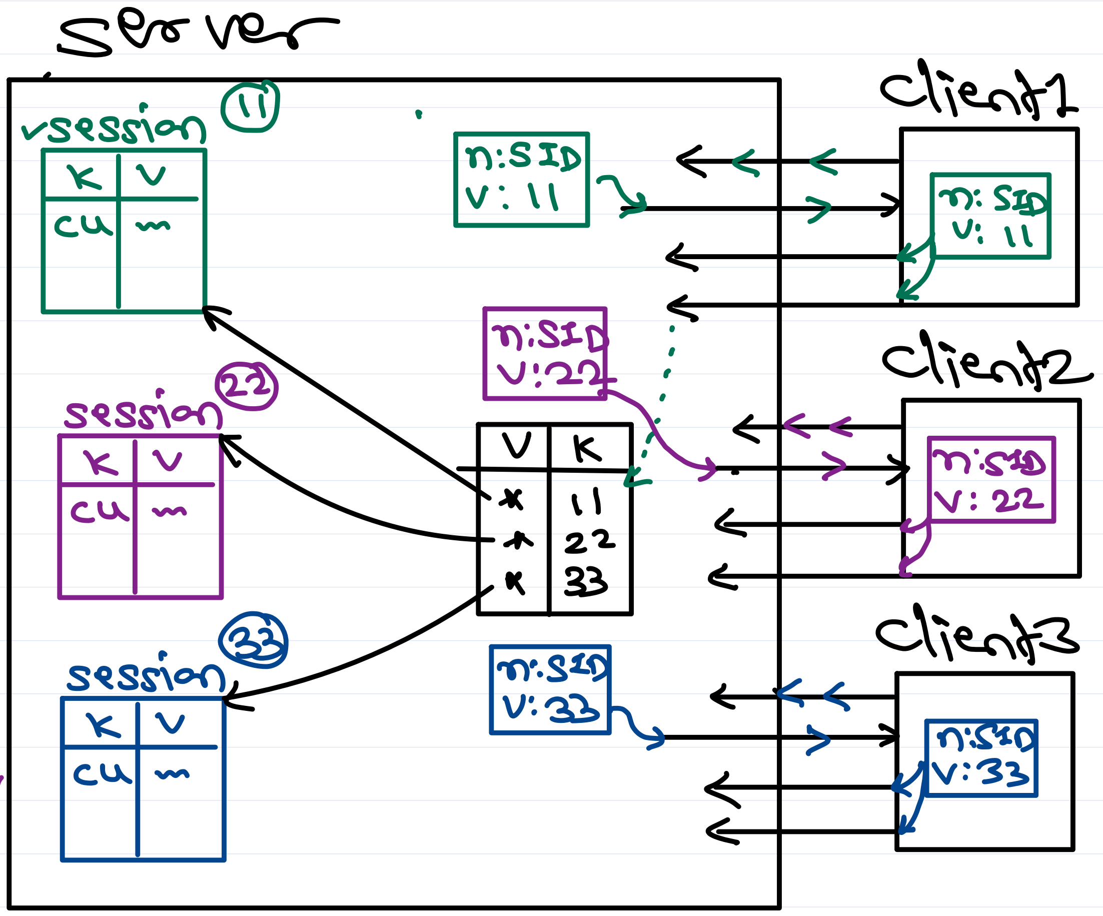
session.setAttribute("key", value);

To retrieve key-value from session:

value = session.getAttribute("key");

To destroy the session:

session.invalidate();



Session tracking using Cookie (JSESSIONID).

Session

session = req.getSession();

① check if session id cookie is present in current request. If available, find the session object corresponding to that id and return it.

② if session id cookie doesn't exist, create a new session and save it with new session id (in server internal mem). Create a new session id cookie and send it to client with next response.

Session tracking modes:

web.xml

```
<session-config>
    <tracking-mode> COOKIE </tracking-mode>
                     or
                    URL
</session-config>
```

Session tracking by URL rewriting

http://lh:8080/app/url

after re-writing

http://lh:8080/app/url;jsessionid=xxxxx

programmer need to rewrite each url in appln (form action, a href, resp.sendRedirect, ...) using

① reurl=resp.encodeURL(url);

OR

② reurl=resp.encodeRedirectURL(url);

QueryString

when req next page, the page url can be appeded with additional key-value pairs.

It is called as query string.

http://lh:8080/app/page?key1=val1&key2=val2

This data is accessible in that page as req. parameters.

```
String val1=req.getParameter("key1");
String val2=req.getParameter("key2");
```

Req parameters vs Req attributes

1. received from client.
2. usually data from form controls or query string.
3. always String.
4. to access:

```
String v = req.getParameter("_");
or String[] v =
    req.getParameterValues("_");
```

1. sent from servlet1 to servlet2
2. any data added by servlet1.
3. any java Object
4. servlet 1:

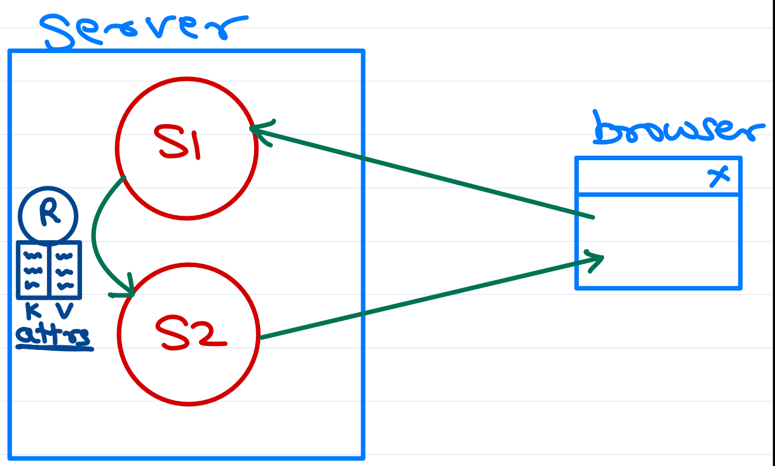
```
req.setAttribute("k", val);
servlet2:
val = req.getAttribute("k");
```

5. Used only while forwarding / including request.

* both are destroyed when req processing is Completed i.e. response is generated

Attributes - Scopes

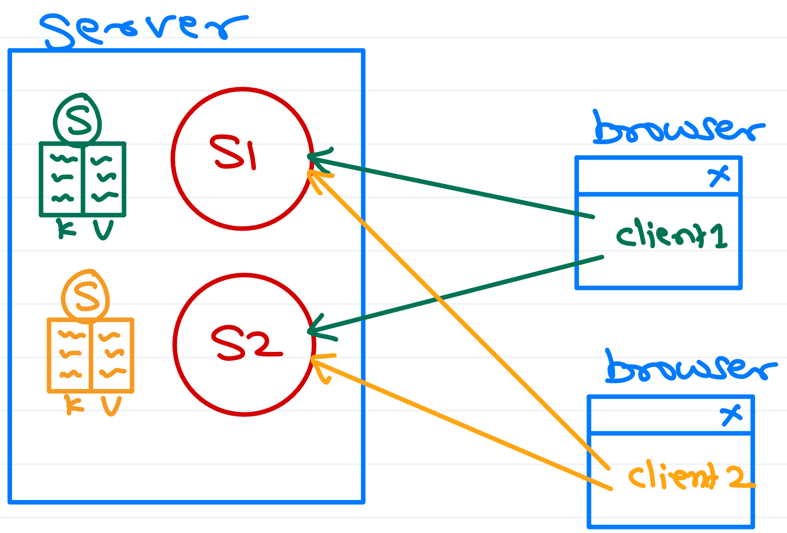
Request attributes



Req attributes will live only for current request - being forwarded or included.

"req" obj is arg to doGet(), doPost(),... methods.

Session Attributes



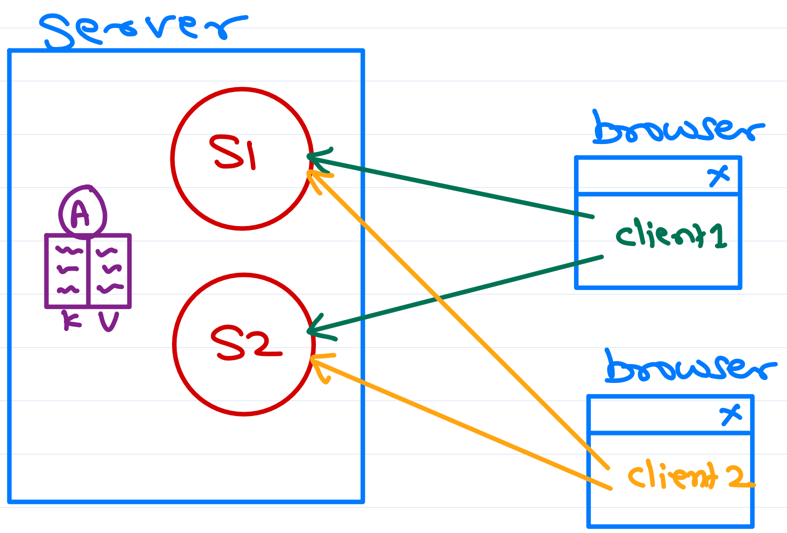
Session attributes are accessible across all requests for current user.

Separate session for each user.

To get current user session:

HttpSession session = req.getSession();

Servlet Context attributes



ServletContext - application object - one for whole appln.

ServletContext attributes are accessible across all requests for all users.

To get appln/ServletContext:

ServletContext appln = req.getServletContext();

SUNBEAM

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>