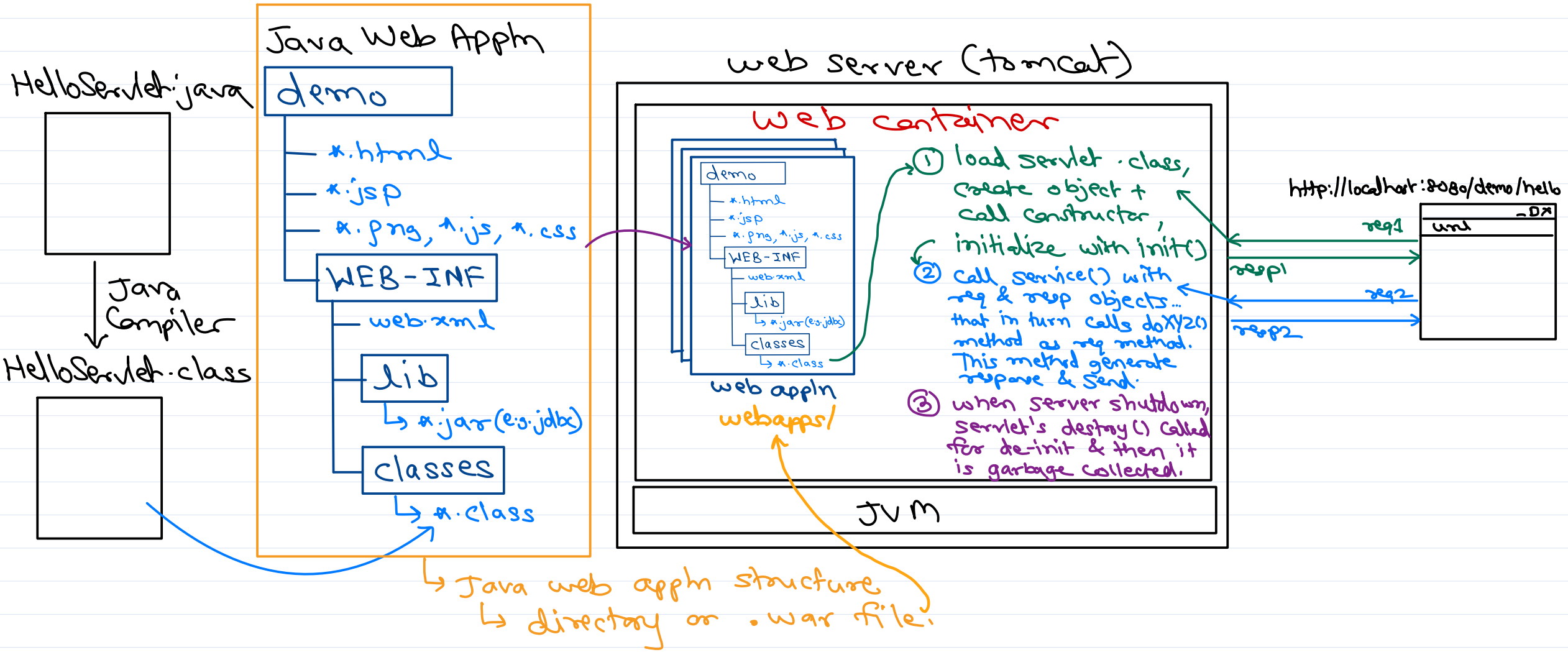
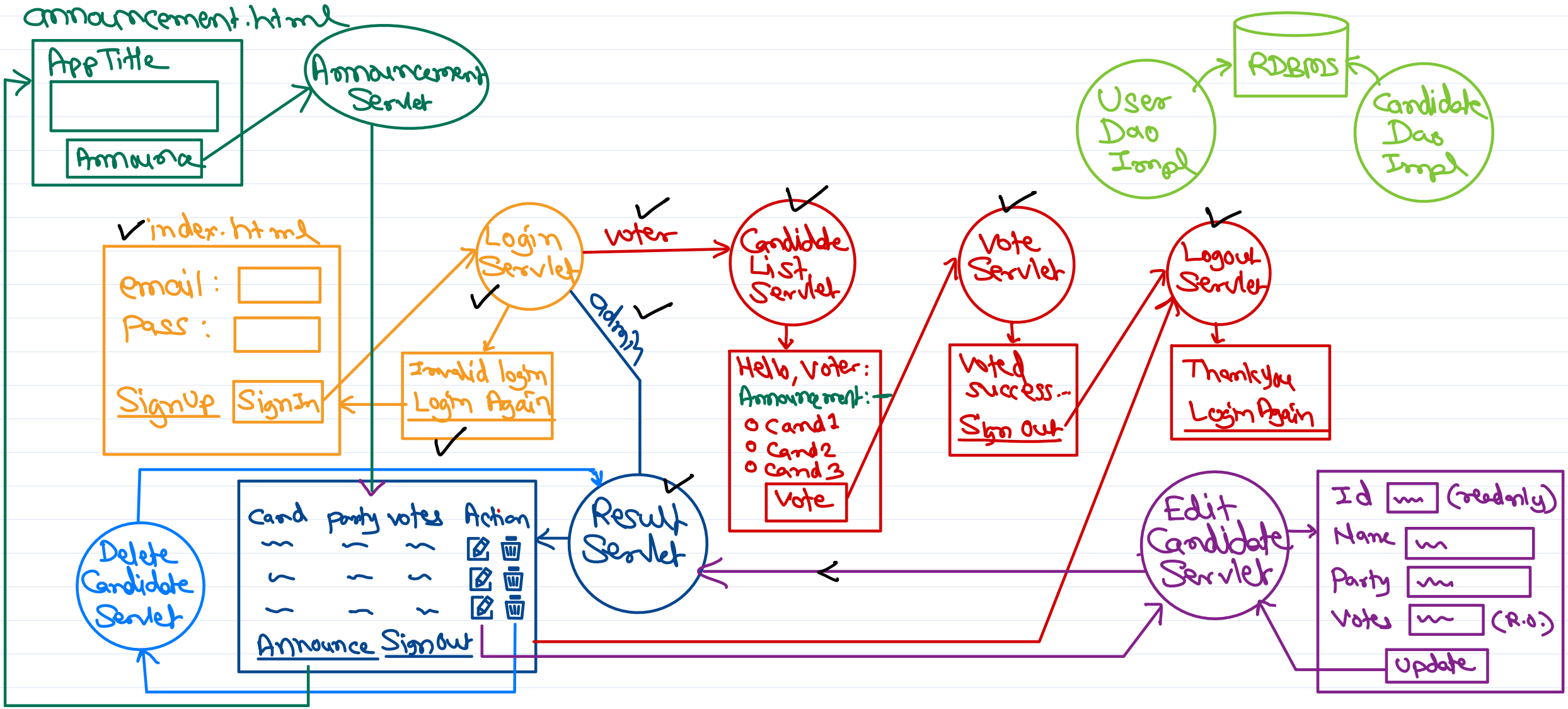
Advanced Java

Trainer: Nilesh Ghule

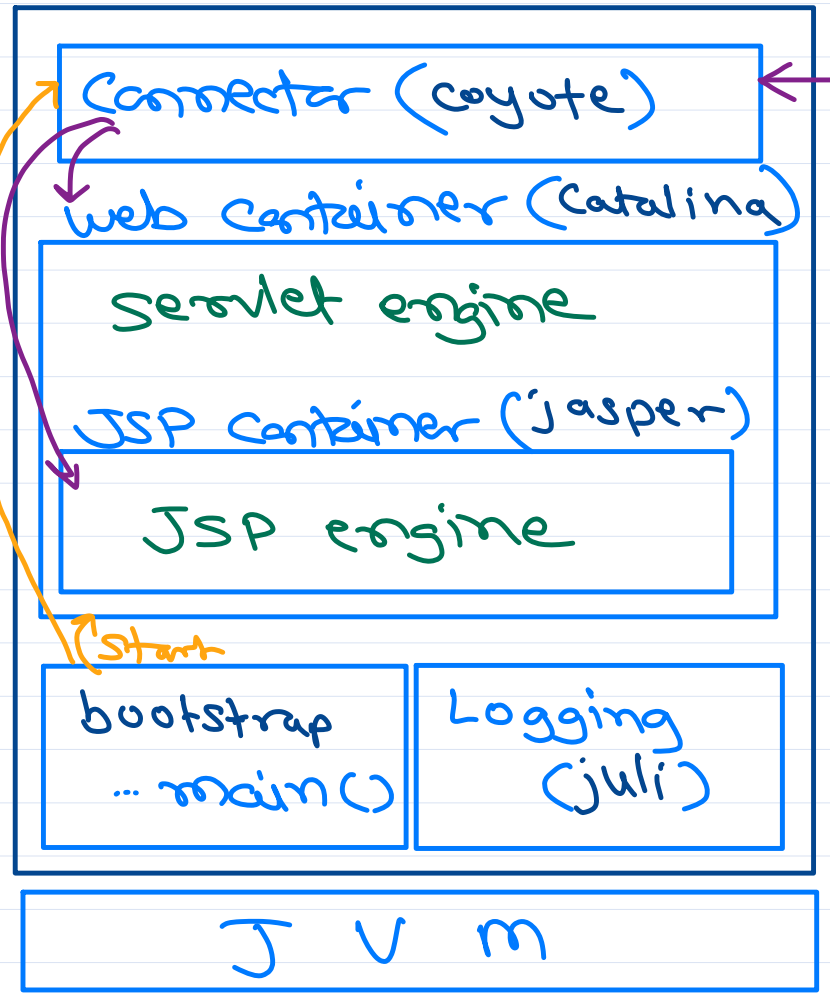
HelloServlet execution



Election Management



Tomcat architecture



req

Key parts:

1. Connector (accept reqs)
2. web container (req processing)
3. web applns running in tomcat.

Request Processing

1. accept request
2. url mapping
   - @WebServlet or web.xml
3. servlet object
   - created on 1st req
   - accessed from pool on next req
4. Response generation
   - doXyz() method
5. HTTP response

Thread model

- tomcat creates a collection of threads during startup - executor thread pool.
- on each req to a connector, a thread from pool is assigned to process that request.
- upon completion, thread returns back to pool.
- max num of threads in pool can be configured in server.xml.

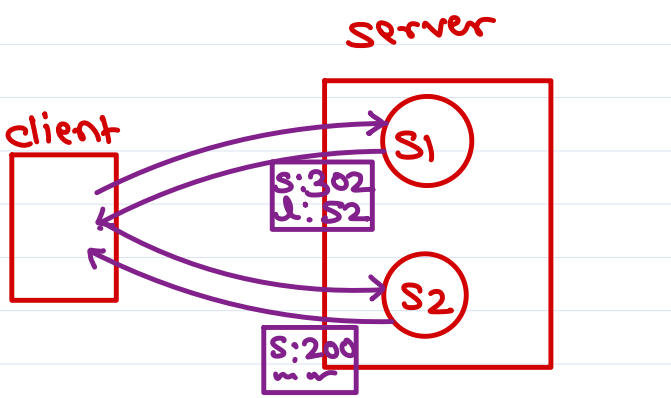
Request parameters

data from prev page controls or query string will come with req object.
It can be accessed in next servlet using.

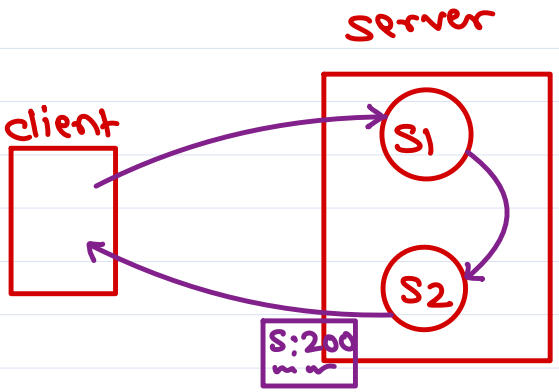
```
String val = req.getParameter("param-name");

String[] vals = req.getParameterValues("param-name");
```

- vals → checkbox, listbox,..
- val → textbox, radio, dropdown,

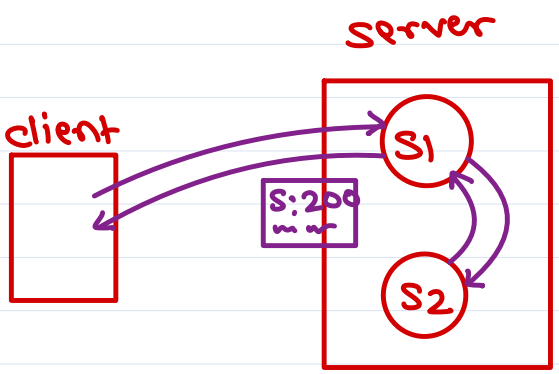
Inter-Servlet Communication



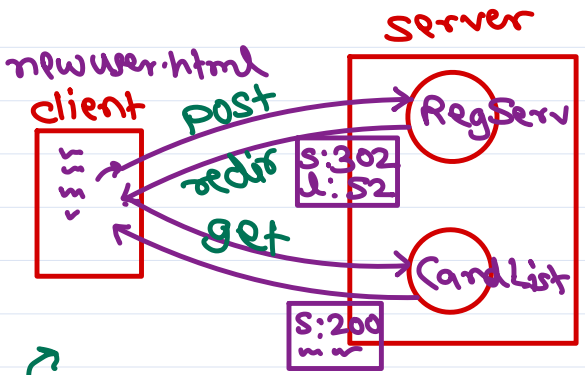
resp.sendRedirect("s2");



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

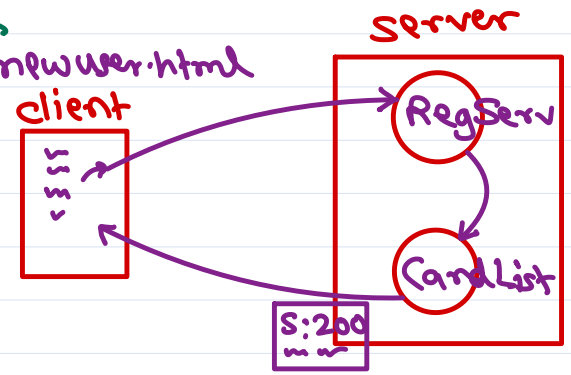


rd = req.getRequestDispatcher("s2");
rd.include(req,resp);



PRG pattern.

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

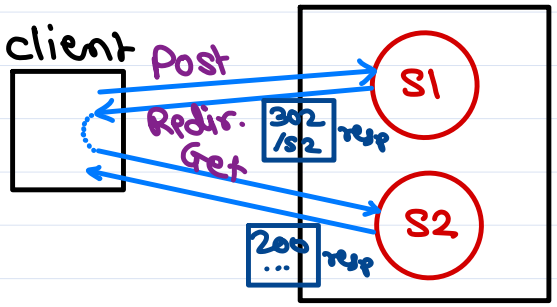


if client refresh, only last req (GET) is replayed & data fetched again. Earlier data not posted twice.

Servlet Communication/Navigation

Http Redirection

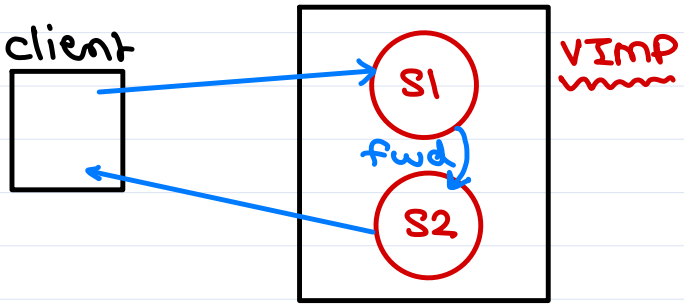
```
resp.sendRedirect("url");
```



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.

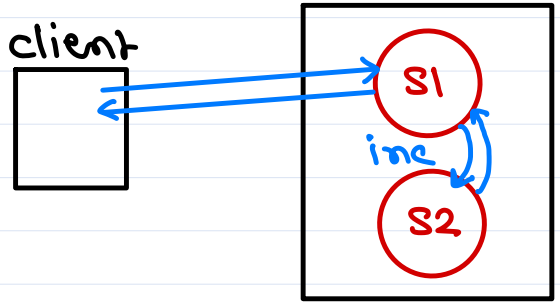
forward() ← RequestDispatcher → include()

```
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
  - QueryString
  - Hidden fields
  - ~~HTML Storage~~

  ✓ Need less server resources.
  x Visible to client
  * Client can tamper

* Server side state mgmt
  - Session
  - Request
  - Application a.k.a. ServletContext

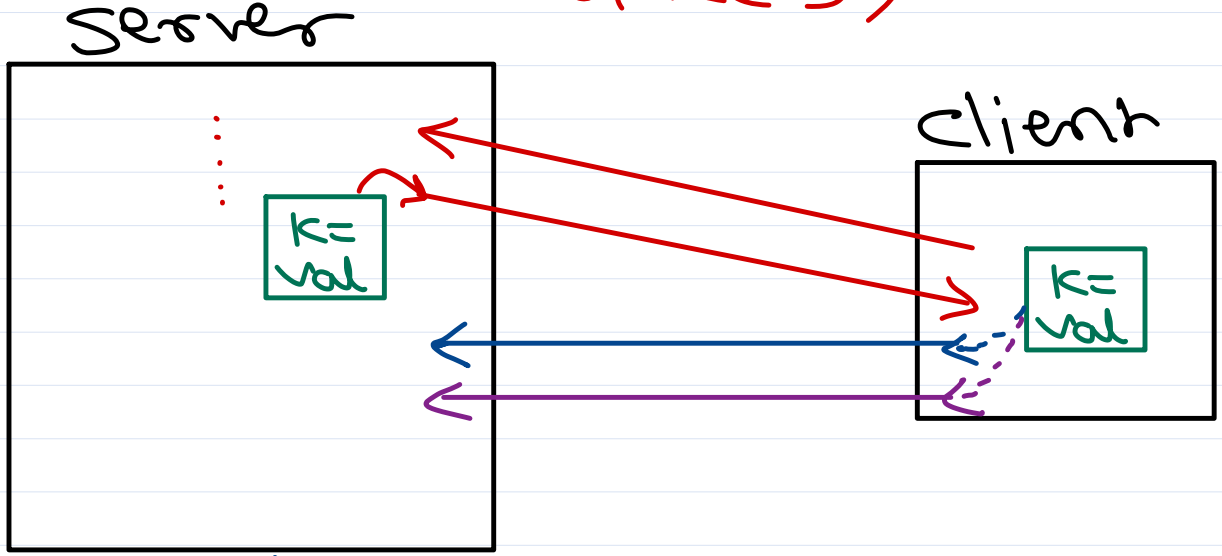
  ✓ Secure (Not accessible to Client)
  x Need more server resources.

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.

* 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;
    }
}
```

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>