Advanced Java

Trainer: Nilesh Ghule

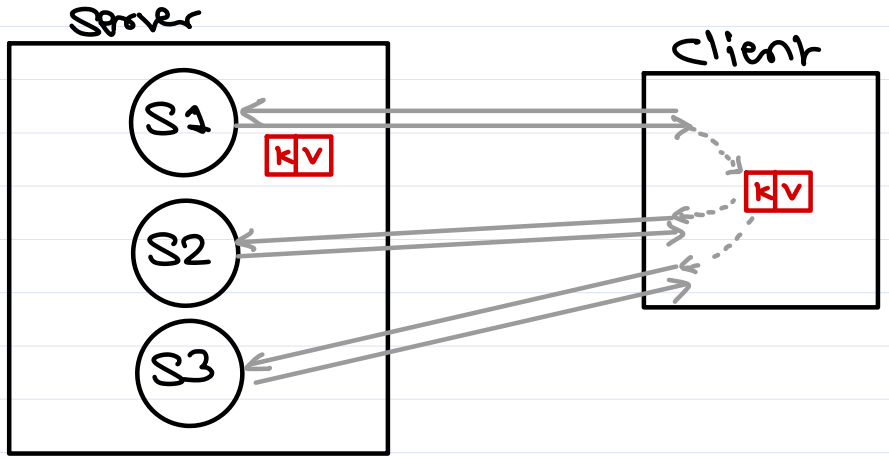
State Management

State Management = Maintaining client state/information/data across multiple requests.

- Client side State mgmt
  - ✓ less data on server (RAM)
  - ✗ not secure (visible to client, can be modified by client)
  1. Cookie
  2. query string
  3. hidden form fields
  4. session/local storage (JS only)
- Server side State mgmt
  - ✗ more data in server RAM
  - ✓ secure
  1. Session
  2. request
  3. application/ServletContext

Cookie

Cookie = text (String) → key-value pair
= max size = 4 KB
= stored on client side by server/applⁿ
↳ browser memory = temp cookie
↳ client disk = permanent cookie



Cookie usage

① server create cookie & send to client via response.

```
Cookie c=new Cookie("key","value");
c.setMaxAge(seconds); // for persistent cookie.
resp.addCookie(c);
```

② with each request client send cookie back to server via request.

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

③ To destroy the cookie

```
Cookie c=new Cookie("key","");
c.setMaxAge(-1);
resp.addCookie(c);
```

Cookie

name = key
value = val
domain = server/host
path = webapplⁿ
expire = dd:tim/sec:im
secure = true/false

HttpSession

HttpSession is like a hashtable for each user/client in server memory (RAM).

String Object

key	value
~	~
~	~
~	~

} session attributes

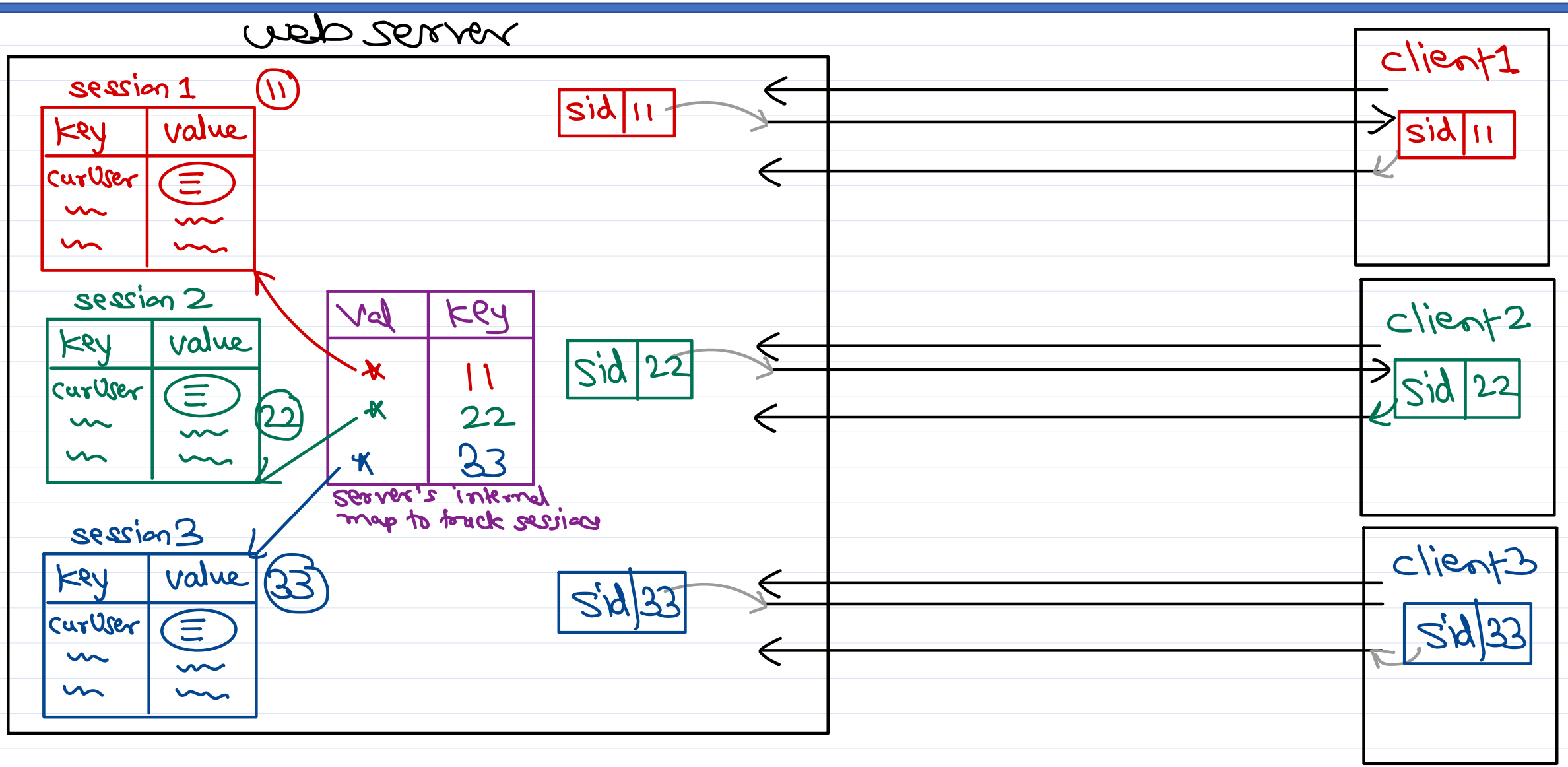
```
session = req.getSession();
```
- get session of current user.
- if no session is avail, create new session for that user.

```
session.setAttribute("key", value);
```
- add an attribute into session

```
value = session.getAttribute("key");
```
- retrieve the attribute of given key.

```
session.invalidate();
```
- destroy the session of current user.

Session Tracking



Session Tracking

Cookies can be disabled in browser. In this case session will not work.

Session tracking can be alternatively done using url rewriting.

http://localhost:8080/app/page;jsessionid=223344

Every url used in appln needs to be rewritten.

```
encurl = resp.encodeRedirectURL(url);
resp.sendRedirect(encurl);
```

```
encurl = resp.encodeURL(url);
out.printf("<a href='%s'>Label</a>", encurl);
```

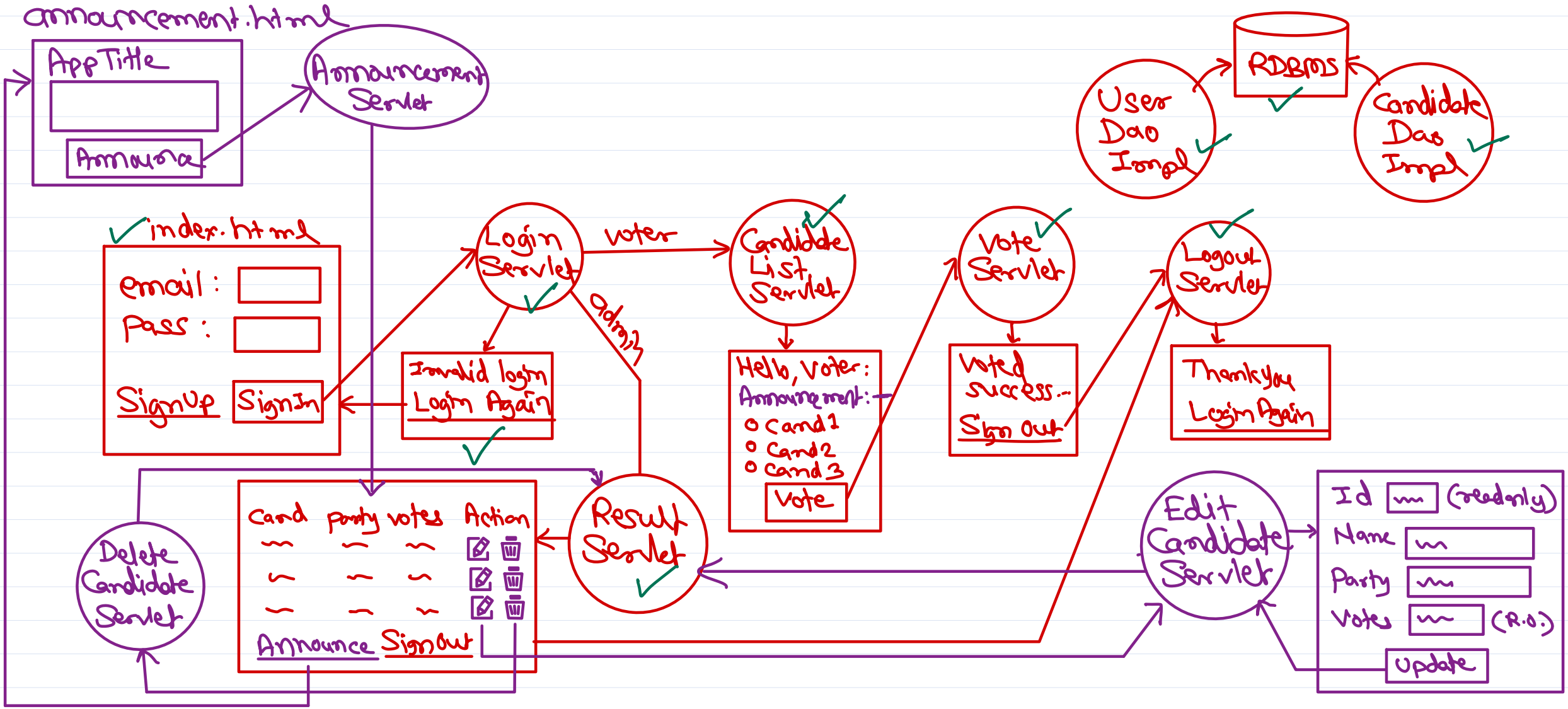
```
encurl = resp.encodeURL(url);
out.printf("<form ... action='%s'>", encurl);
```

* change tracking-mode into web.xml

default: COOKIE

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

Election Management



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 queryString
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:

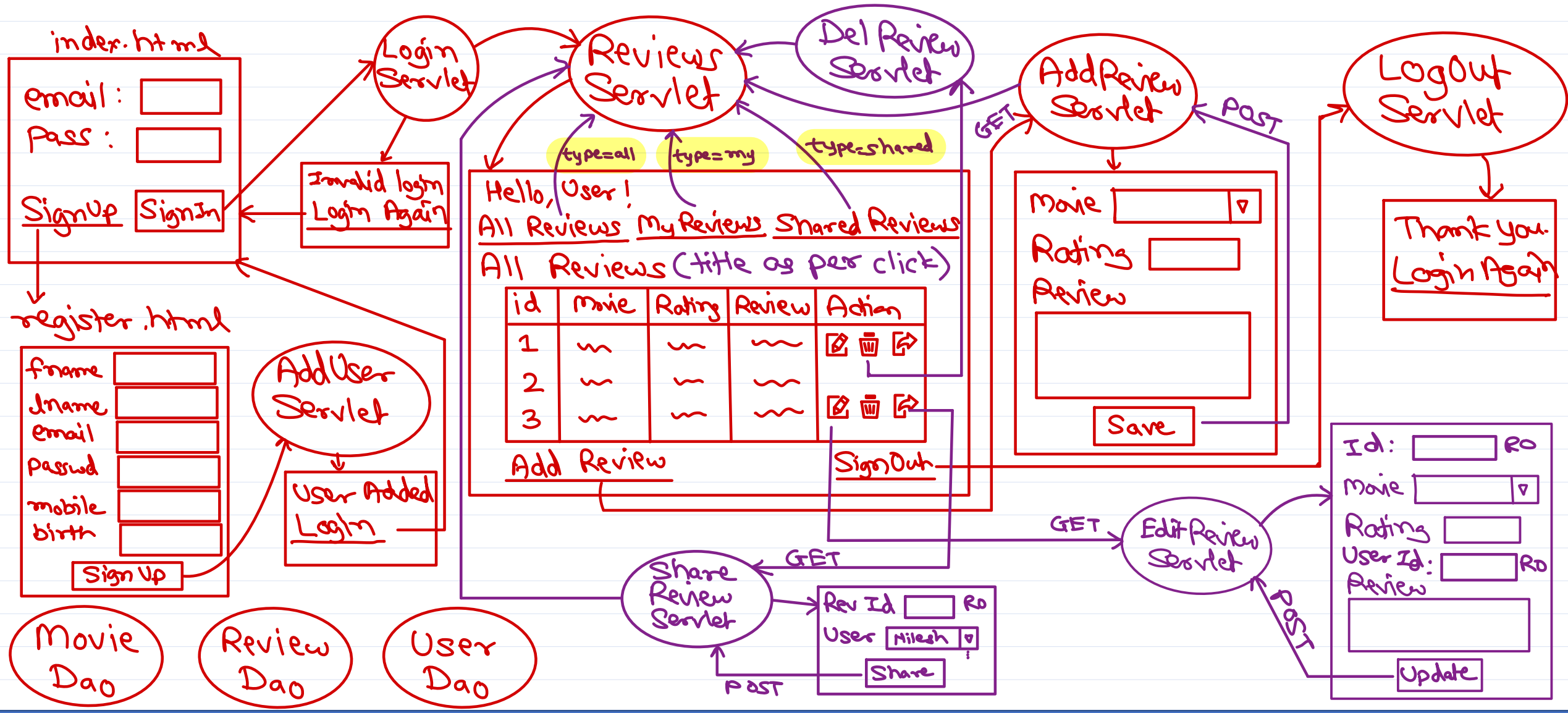
```
req.setAttribute("key", value);
                        ↓
                      Object
```

3. accessed in next servlet:

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

4. Object values
5. until request is completed i.e. response is generated.

Movie Review Assignment



Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>