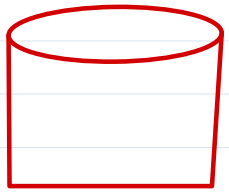
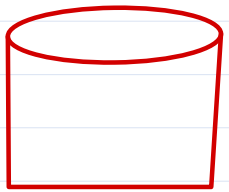
Advanced Java

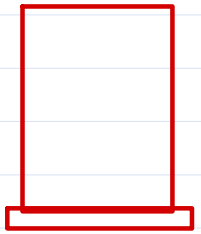
Trainer: Nilesh Ghule

Java EE - Enterprise applications

Java EE is set of specifications/standards.
e.g. JDBC, Servlet, JSP, JSF, EJB, JPA, ...







DAO

- JDBC
- ORM Hibernate
- Spring Data
- NoSQL Drivers

Business Logic

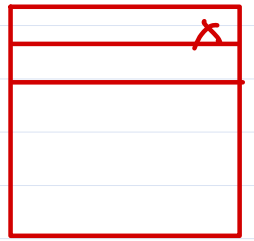
- Servlets
- EJB
- REST Services
- Java Beans
- Spring Beans

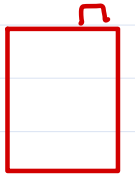
Front End

- JSP
- Thyme leaf
- JSF
- React

Spring MVC

Browser



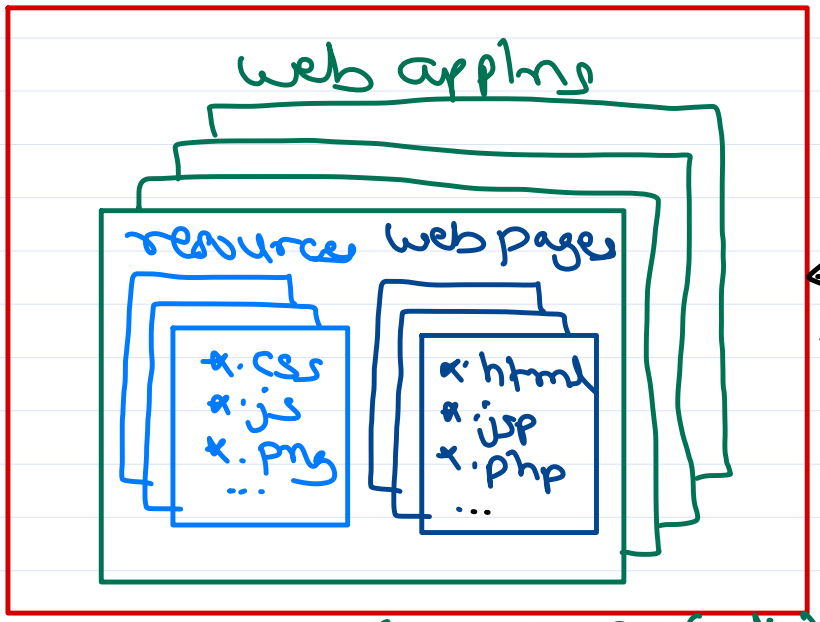


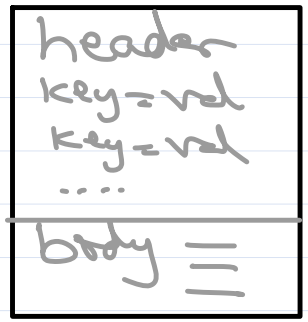
Spring Data	Spring Boot
ORM → JPA → Hibernate	Spring → Core → MVC ...
JDBC | Servlets | JSP	

HTTP protocol

web server | web client | server | client

web server
web appln
web page

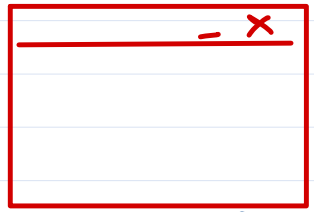




* HTTP is connectionless.
* HTTP is stateless.

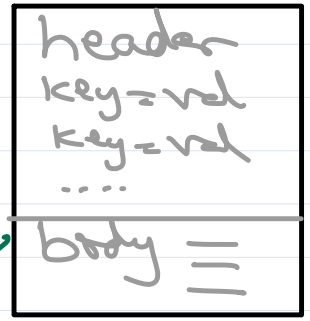
http://server:port/appln/resource?k=v
↳ uri
query string

browser



req

resp



* HTTP → Request Response model.

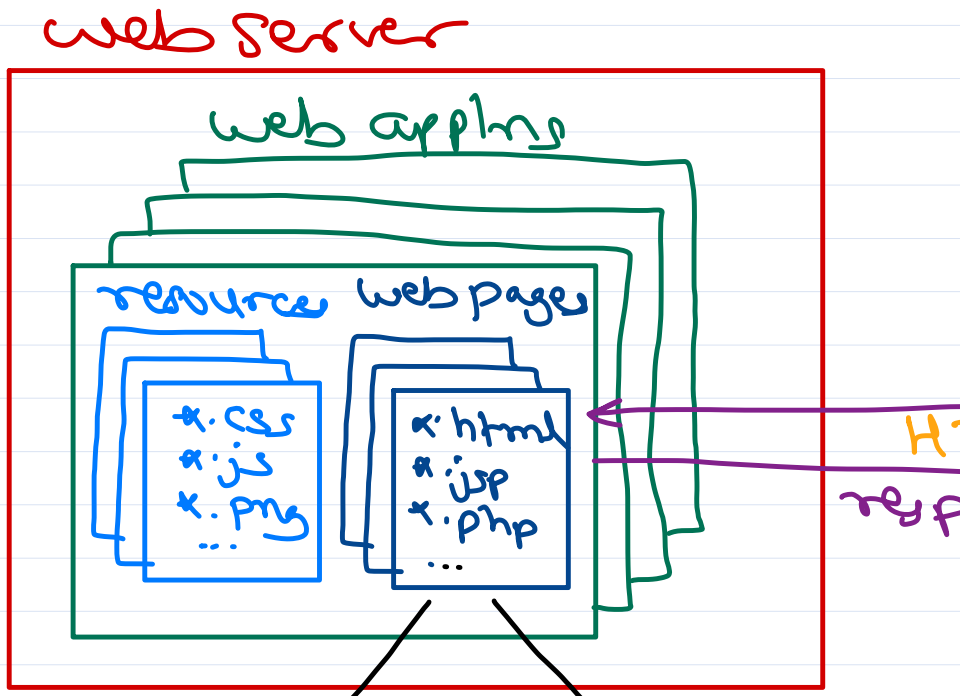
client ⇄ server (pull).
req
resp

resp
① status → 2xx (success), 3xx (redir), 4xx (client error), 5xx (server error).
② content type
③ content length
④ server info
⑤ cookie (also in req).

*req
① content type → text/html*, text/xml, application/json*, image/jpeg, audio/mpeg, application/form-data*
② content length
③ authorization ...
④ client info.
⑤ server+port+uri
⑥ req method → get, post, put, delete, head, trace, options

GET req: ① browser url enter ② hyperlink click
POST req: ① form post method ② form get method ③ after redir.

Web Server



GET

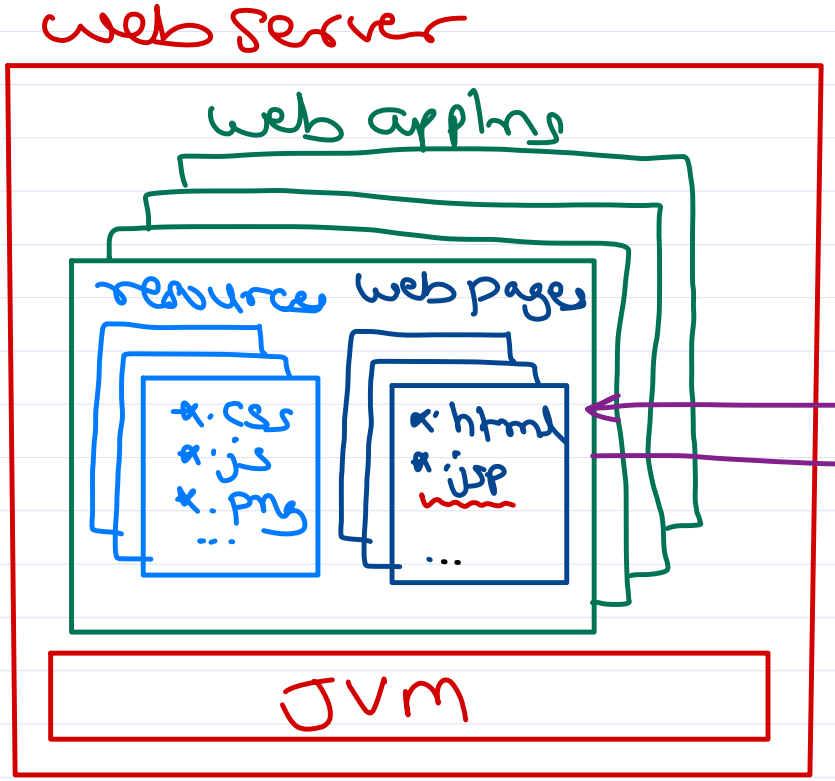
1. go to url in browser.
2. click on hyperlink.
3. http redirection
4. html form with method="get".

POST

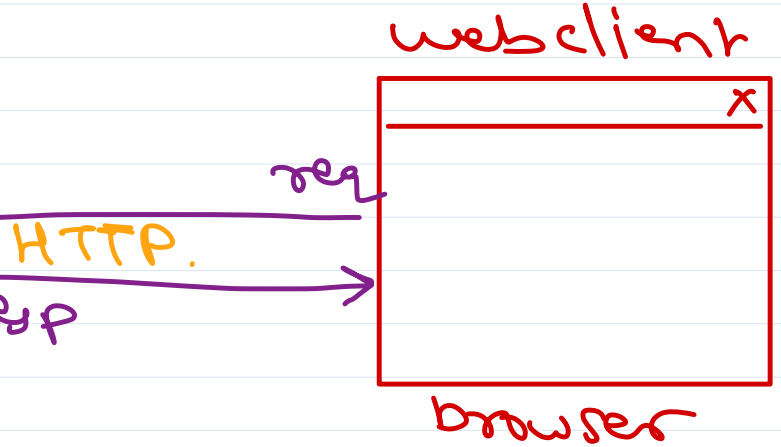
1. html form with method="post".

e.g. IIS, Apache Tomcat, JBoss, ...

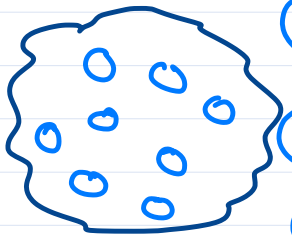
Java Web Server



Standard/specification = set of rules.
- given in form of interfaces, classes & ...
- e.g. JDBC, Java EE.



Java EE is set of specs → servlet + jsp + jsf + jpa + jndi + ejb + ...



4) ...
3) JNDI
2) Connection pool
1) SSL config

Java Web Server = web container + other services
↳ servlets + jsps.
e.g. Tomcat, Jetty, Lotus, ...

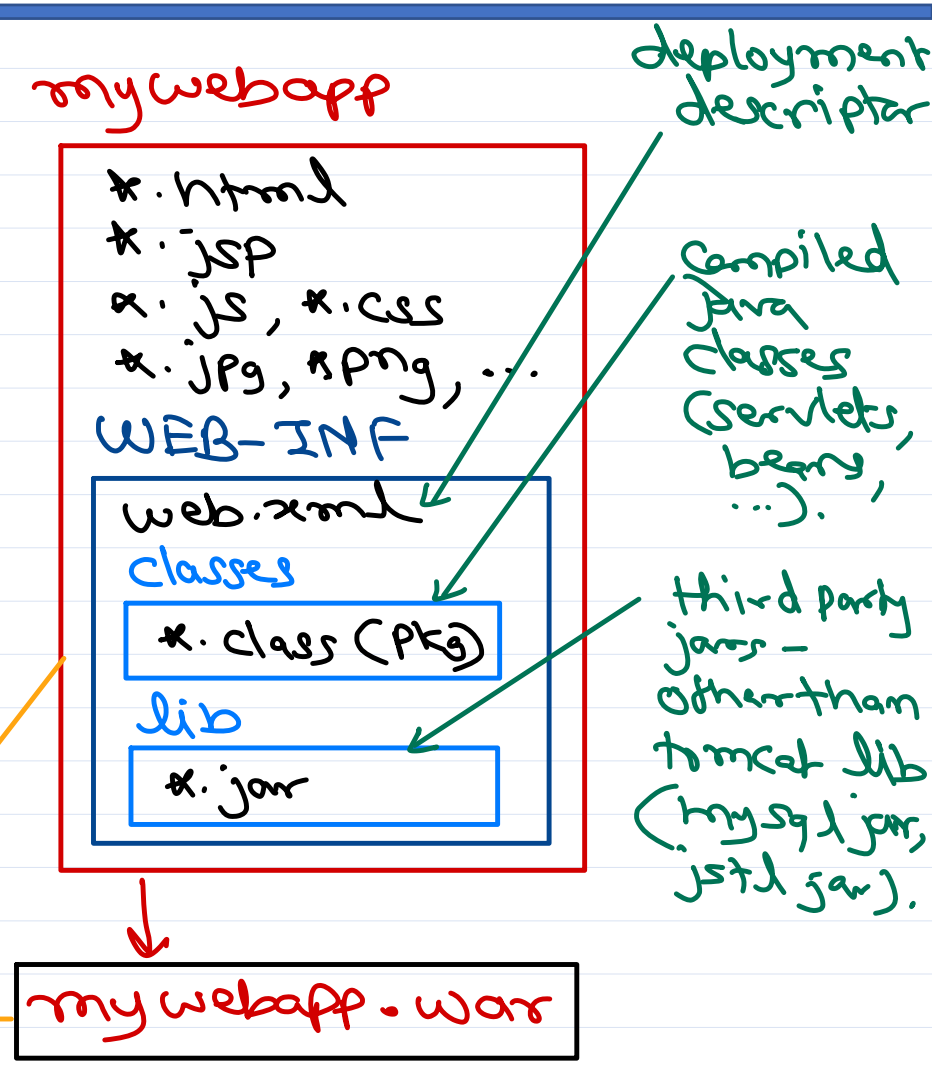
Java App server = web container + other services + ejb container
↳ servlets + jsps. ↳ ejb
e.g. JBoss, web logic, websphere, ...
Glassfish (educational).

Apache Tomcat

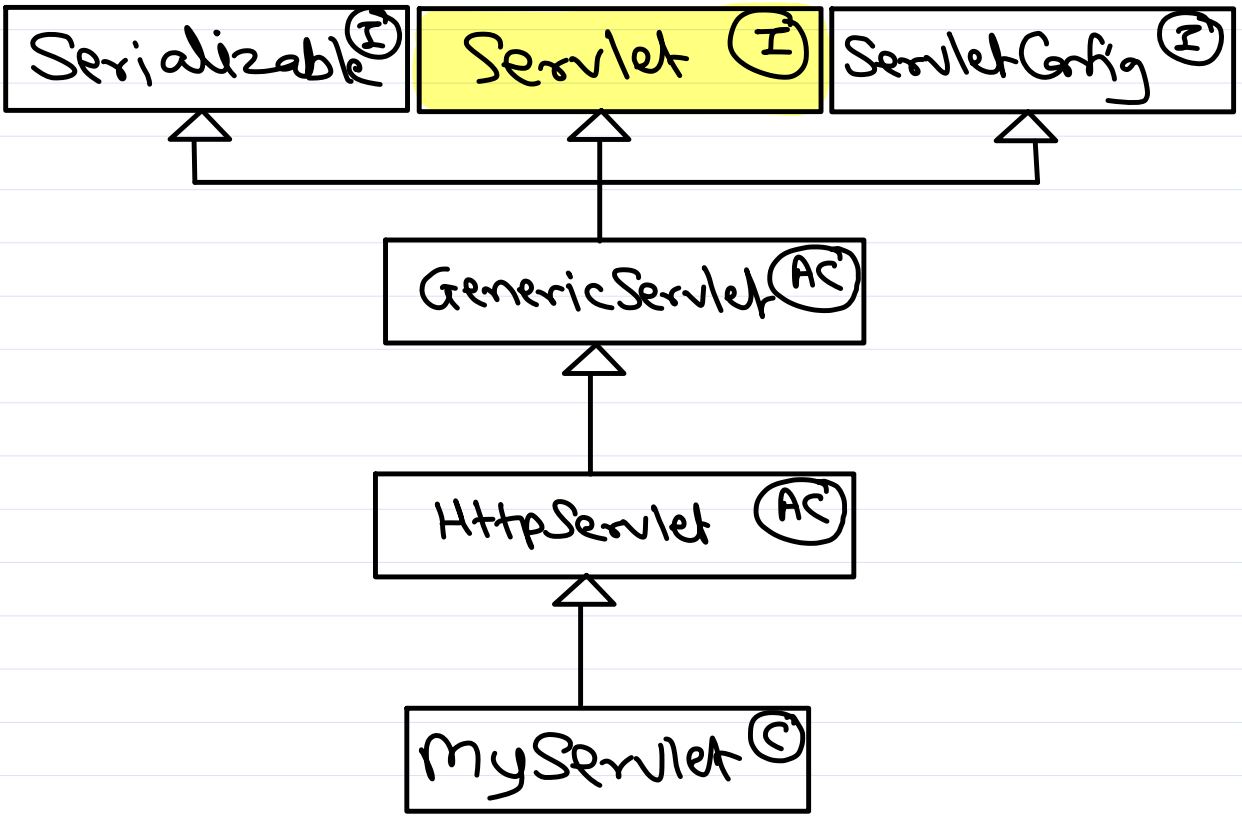
* To start/test tomcat on your system:
  1. download & extract tomcat.
  2. environment var → JAVA_HOME → path of jdk.
  3. tomcat dir → bin → startup.bat ✓
  4. browser → http://localhost:8080
  5. tomcat dir → bin → shutdown.bat ✗

* Tomcat directories
  1. bin : start/stop scripts, tomcat main executable.
  2. conf : config files of tomcat
     e.g. server.xml to set tomcat port (default: 8080)
  3. lib : tomcat libs/jars (which 'implements javaee specs) e.g. servlet-api.jar, jsp-api.jar, ...
  4. logs : web server execution logs (exceptions ...).
  5. temp & work : temp files created at runtime for appln execution.
  6. webapps : Hot deployment directory.
     - applns to be deployed are copied here
     - hot ⇒ applns copied while server is on are auto-deployed.

Java Web Application



Servlet is a java class that is executed in java web server when req is received from client and produces response that is sent back to the client.



Servlet interface:

① init() → arg: ServletConfig ← info/ metadata of servlet
- when 1st req received for a servlet, obj is created and init() called by web container.
- programmer should override, if any one time initialization is to be done. e.g. JDBC connection, ...
- If failed, must throw ServletException. Now servlet reqs will not be processed further.

③ destroy()
- when appln is undeployed or web server shutdown, destroy is called by web container.
- programmer should override, if any one time de-initialization is to be done. e.g. close connection.

② service() →

Java Servlets

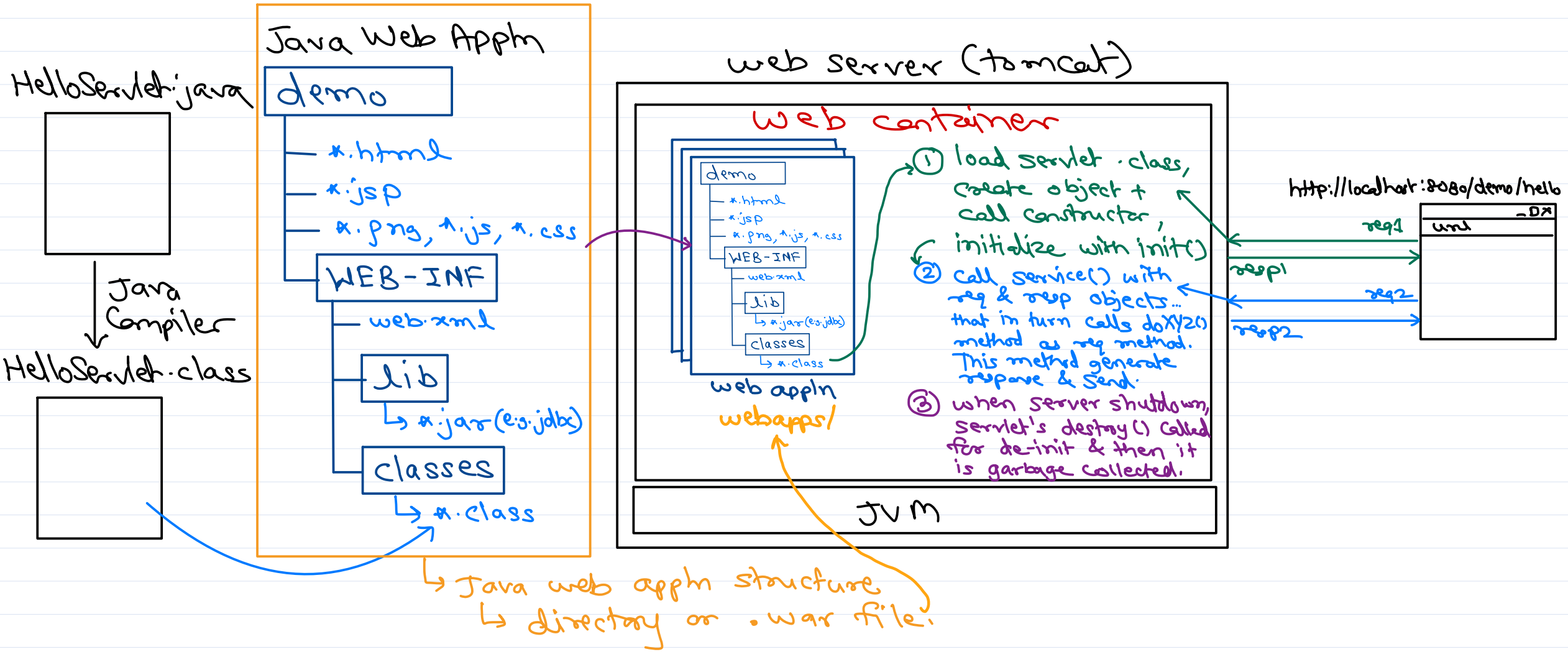
* Servlet interface:
  - @service()
    - called for each req by web container.
    - programmer should override it to process the req and produce the response.

* GenericServlet class
  - provides default impl of init() & destroy().
  - keeps service() abstract.
  * Protocol independent servlet.

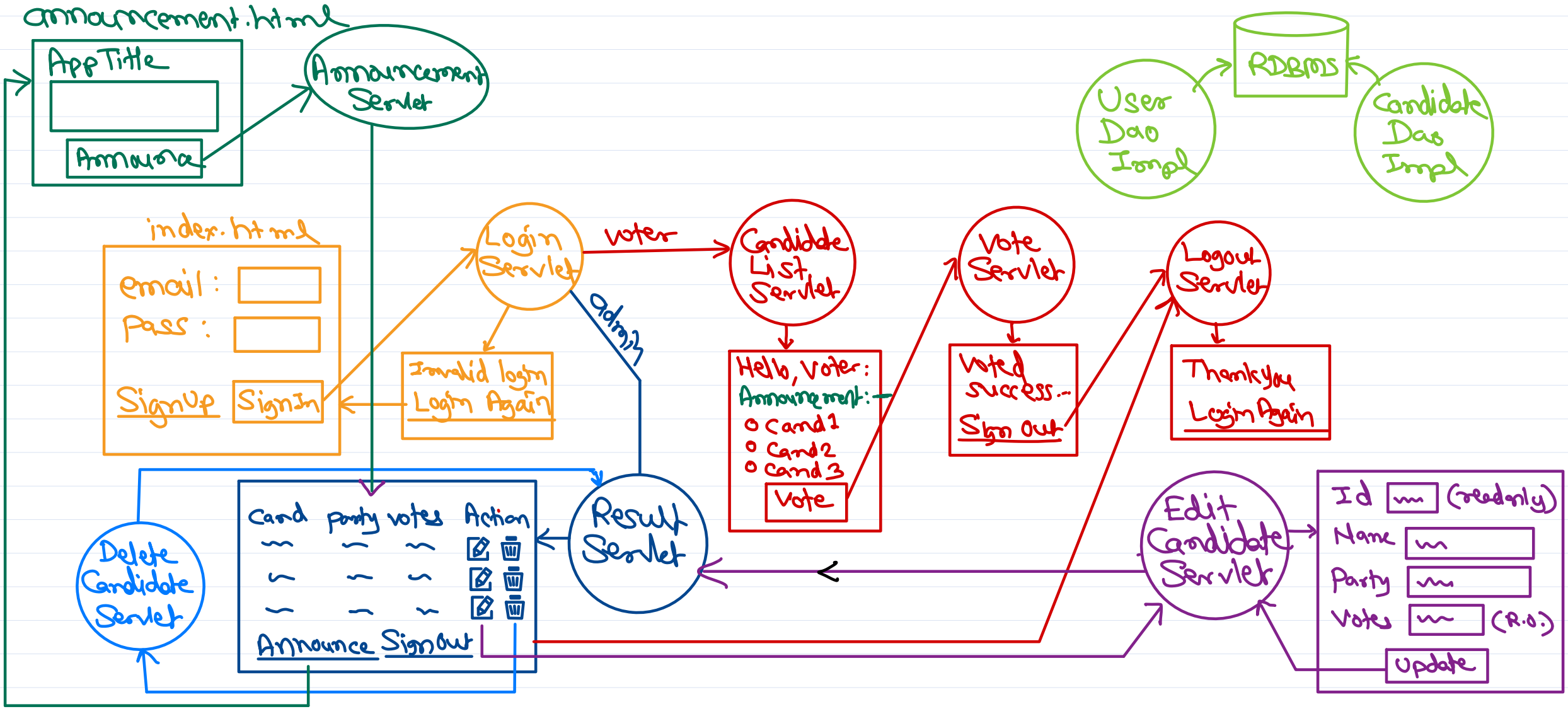
* HttpServlet class
  - class to handle HTTP requests.
  - it overrides service() and internally calls doGet(), doPost(), doHead(), ... methods depend on current req. method.

```
//predefined class
class HttpServlet extends
                    GenericServlet {
@Override
void service(req, resp) ... {
    String m = req.getRequestMethod();
    if(m == "get")
        doGet(req, resp);
    else if(m == "post")
        doPost(req, resp);
    else ...
}
void doGet(req,resp) { throw ... };
void doPost(req,resp) { throw ... };
...
```

HelloServlet execution



Election Management



Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>