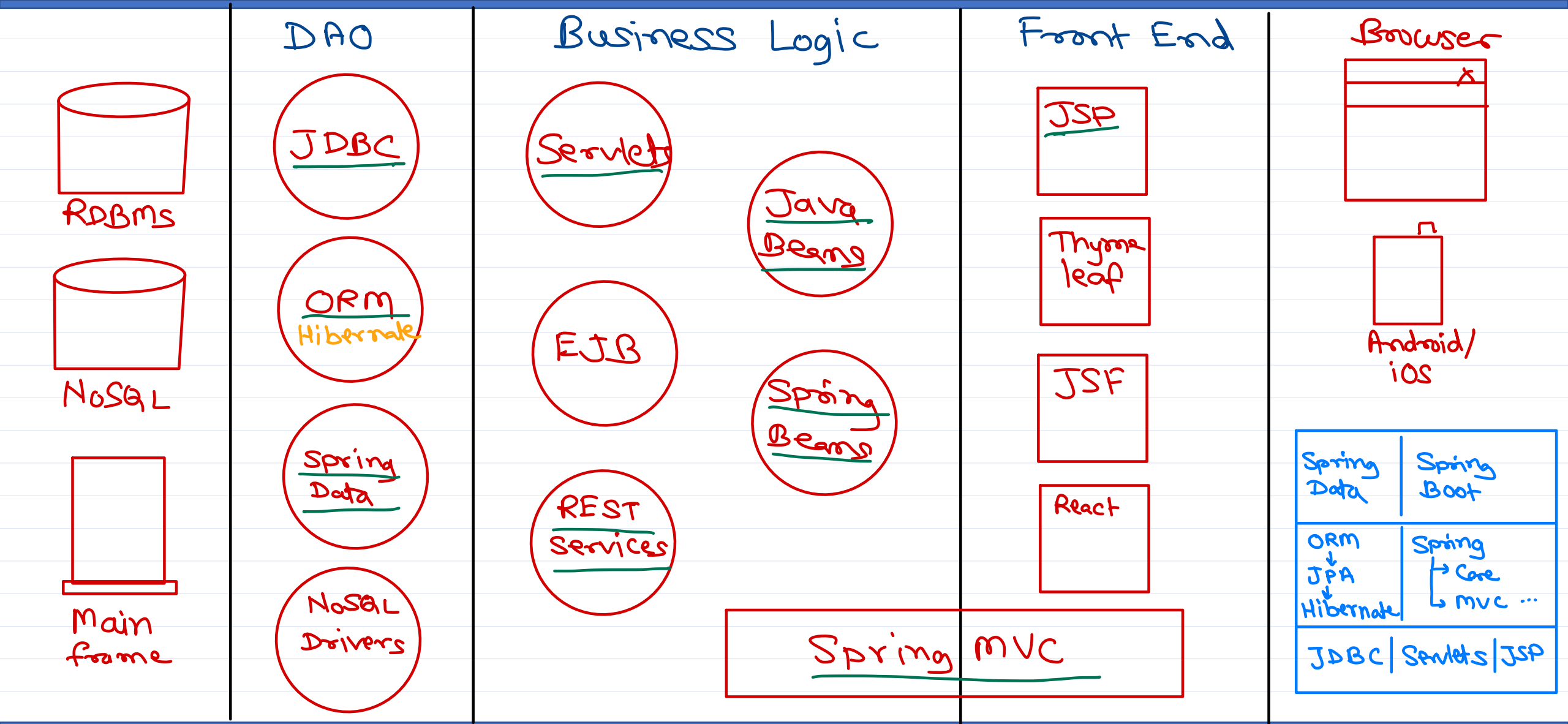# Advanced Java

Trainer: Nilesh Ghule
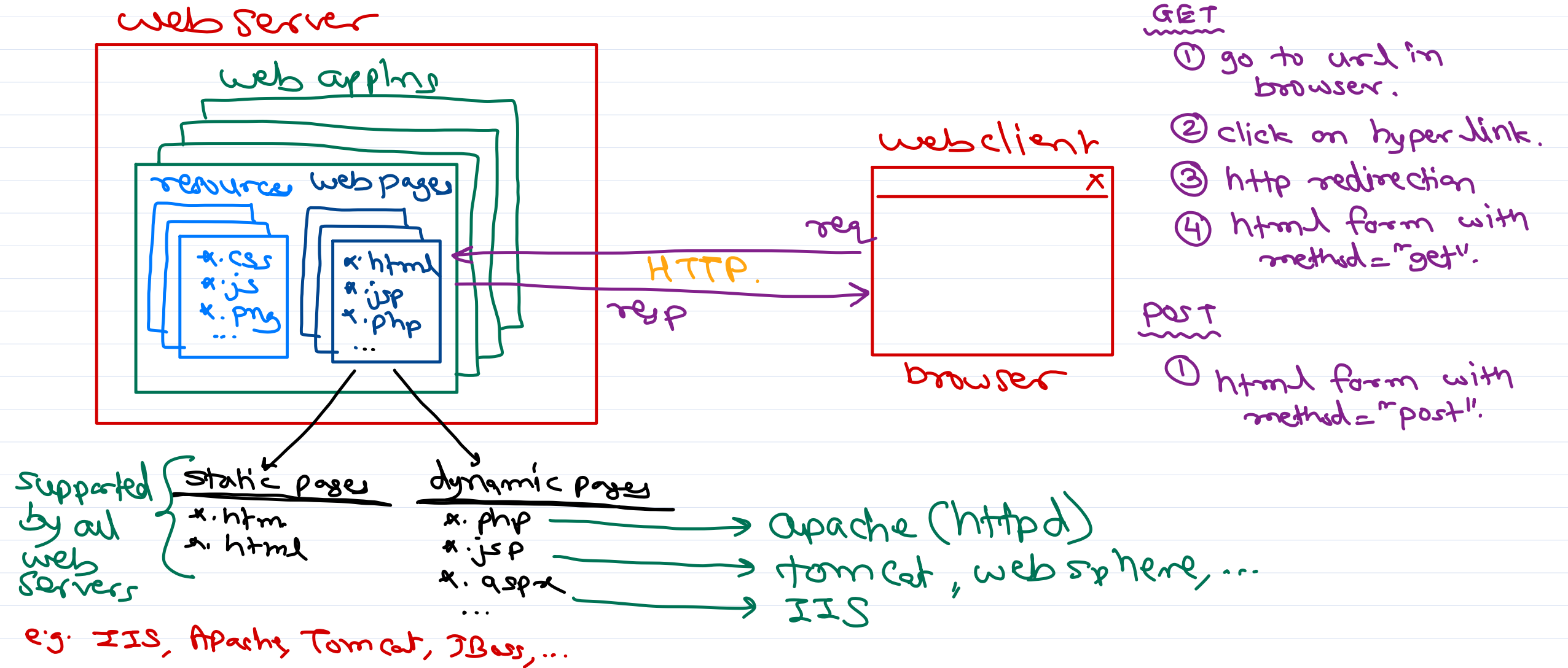
Java EE - Enterprise applications

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

DAO	Business Logic	Front End	Browser

Data / Storage:
- RDBMS
- NoSQL
- Main frame

DAO:
- JDBC
- ORM — Hibernate
- Spring Data
- NoSQL Drivers

Business Logic:
- Servlet
- EJB
- REST Services
- Java Beans
- Spring Beans

Front End:
- JSP
- Thyme leaf
- JSF
- React
- Spring MVC

Browser:
- Android/iOS

Spring Data	Spring Boot
ORM ↓ JPA ↓ Hibernate	Spring ↳ Core ↳ MVC ...
JDBC	Servlets	JSP

HTTP protocol

web server e.g. Apache, tomcat, IIS, ...

web applns

resource   web pages

*.css   *.html
*.js   *.jsp
*.png   *.php
...   ...

```
header
key = val
key = val
....
----------
body  =
      =
```

req →

```
header
key = val
key = val
....
----------
body  =
      =
```

resp

* HTTP is Connectionless.
* HTTP is Stateless.

http://server:port/appln/resource?k=v
                   └→ uri   ↑
browser   Query String

```
┌──────────────┐
│        _ ✗   │
│ ─────────    │
│              │
│              │
└──────────────┘
```

* HTTP → Request Response Model.
  Client ⇄ req → Server (Pull).
         ← resp

resp
→ 2xx (success), 3xx (redir),
  4xx (client error),
  5xx (server error).

① Status
② Content. type   ③ Content length
④ Server info
⑤ Cookie (also in req).

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

Web Server

web server

web applns

resource   web pages

*.css
*.js
*.png
...

*.html
*.jsp
*.php
...

HTTP.

web client

req

resp

x

browser

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".

static pages   dynamic pages
*.htm   *.php ――――→ Apache (httpd)
*.html   *.jsp ――――→ tomcat, websphere, ...
                      *.aspx ――――→ IIS
                      ...

supported by all web servers

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

Java Web Server

java web server ⟵

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

web applns

resource web pages

*.css
*.js
*.png
...

*.html
*.jsp
...

JVM

web client

x

req

HTTP.

resp

browser

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).

* To start/test tomcat on your system:
① download & extract tomcat.
② enviorment 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, tomat main executable.
② conf : config files of tomcat
     e.g. server.xml to set tomcat port (default: 8080)
③ lib : tomat 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.

mywebapp

*.html
*.jsp
*.js, *.css
*.jpg, *png, ...

WEB-INF

web.xml

classes

*.class (pkg)

lib

*.jar

OR

mywebapp.war

deployment descriptor

Compiled Java classes (servlets, beans, ...).

third party jars - other than tomcat lib (mysql jar, jstl jar).

Tomcat architecture

Tomcat architecture

```
Tomcat architecture
┌─────────────────────────────────────┐
│  ┌───────────────────────────────┐   │   req
│  │  Connector (coyote)           │◄──┼──────
│  └───────────────────────────────┘   │
│  Web Container                        │
│  ┌───────────────────────────────┐   │
│  │  Servlet engine               │   │
│  │      (catalina)               │   │
│  │  JSP Container ('jasper')     │   │
│  │  ┌─────────────────────────┐  │   │
│  │  │   JSP engine            │  │   │
│  │  └─────────────────────────┘  │   │
│  └───────────────────────────────┘   │
│  ┌Start──────────┐ ┌───────────────┐ │
│  │ bootstrap     │ │ Logging       │ │
│  │ ... main()    │ │   (juli)      │ │
│  └───────────────┘ └───────────────┘ │
└─────────────────────────────────────┘
┌─────────────────────────────────────┐
│            J V M                      │
└─────────────────────────────────────┘
```
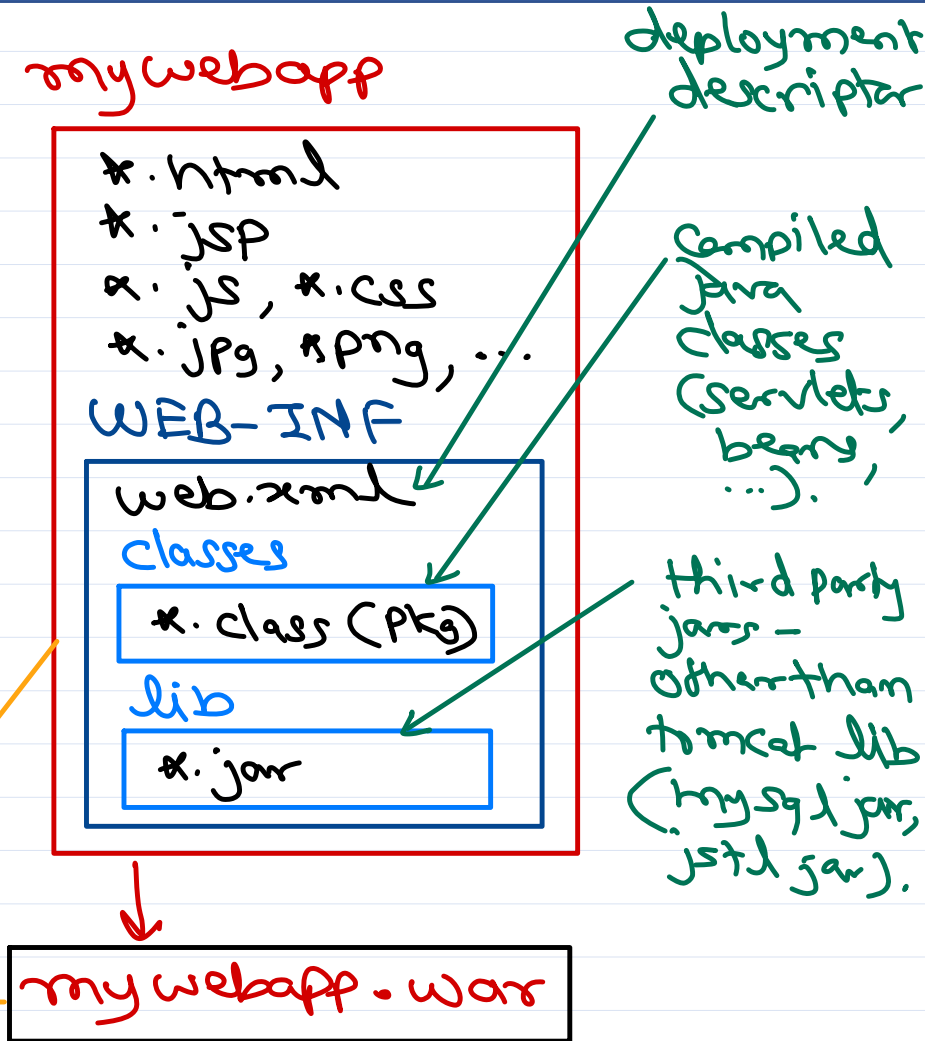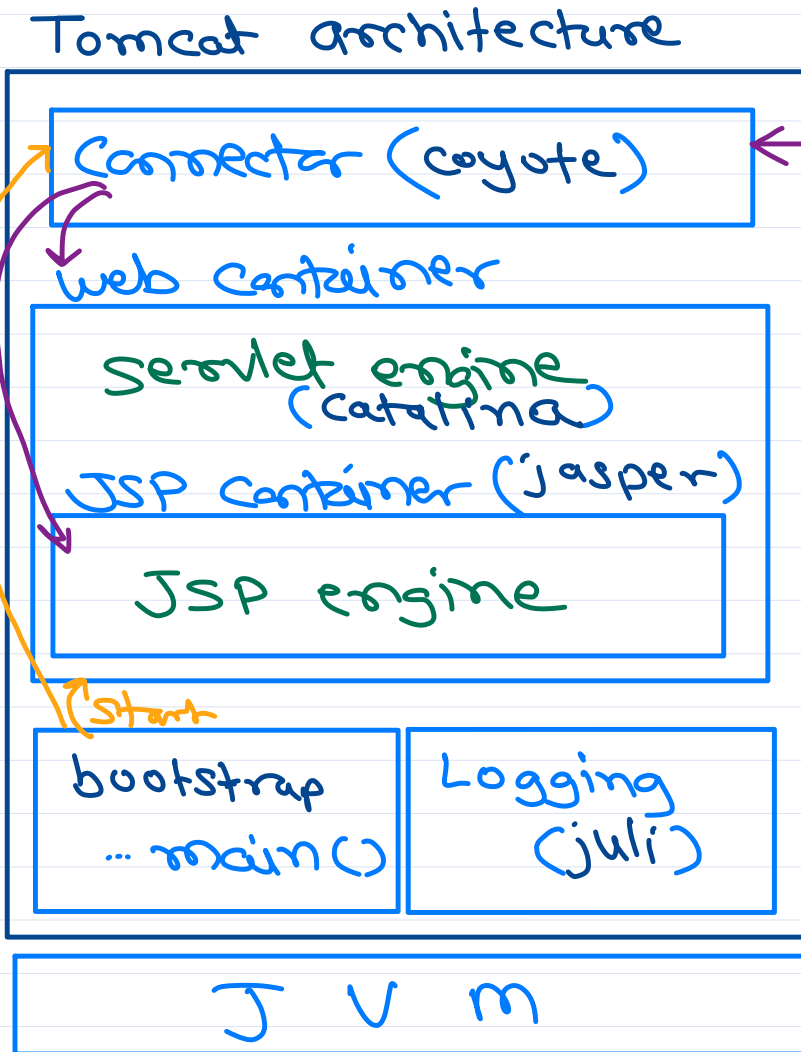
Key parts:
1. Connector (accept reqs)
2. web Container (req processing)
3. web applns running in tomcat.

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
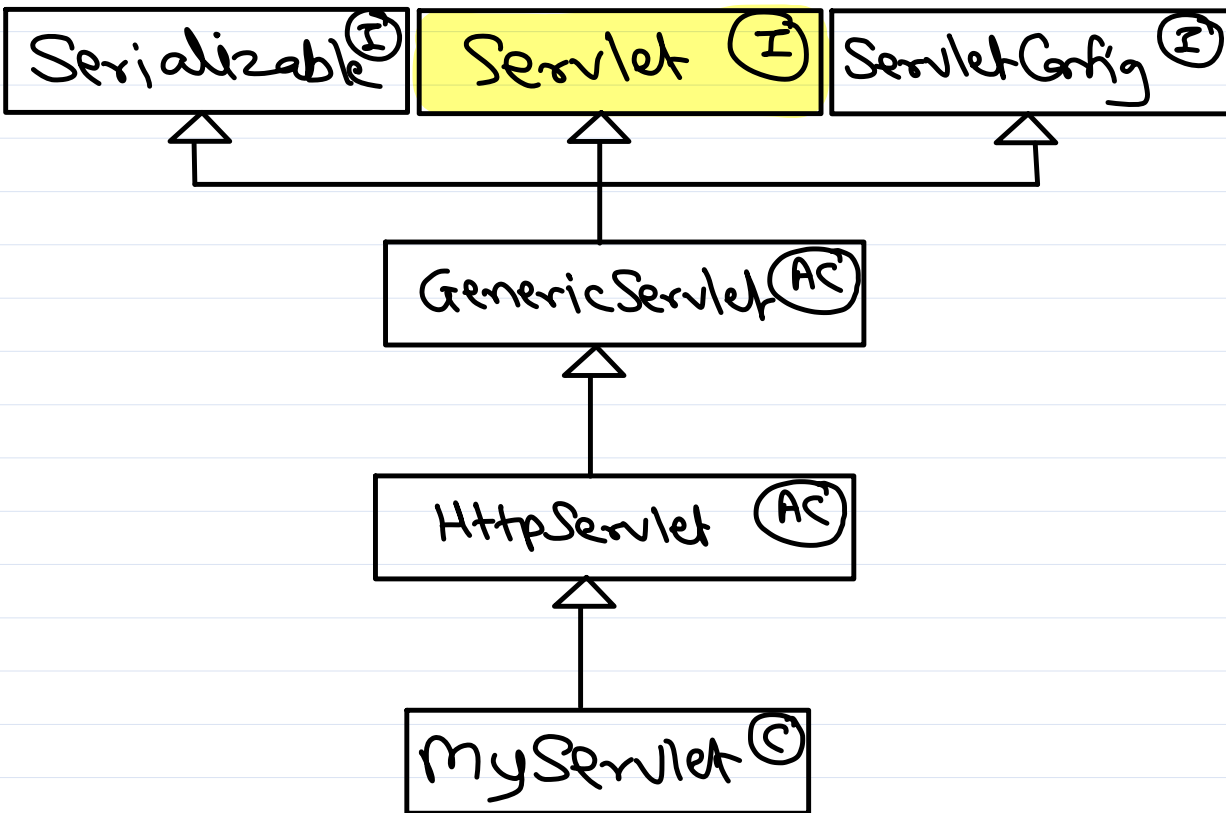4. Response generation
   ↳ doXyz() method
5. HTTP response

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.

Java Servlets

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.

```
┌──────────────┐   ┌──────────────┐   ┌──────────────┐
│ Serializable Ⓘ│   │  Servlet  Ⓘ  │   │ ServletConfig Ⓘ│
└──────────────┘   └──────────────┘   └──────────────┘
        △                 △                  △
        └─────────────────┼──────────────────┘
                          │
                 ┌──────────────────┐
                 │ GenericServlet ⒶⒸ │
                 └──────────────────┘
                          △
                          │
                 ┌──────────────────┐
                 │  HttpServlet  ⒶⒸ │
                 └──────────────────┘
                          △
                          │
                 ┌──────────────────┐
                 │  MyServlet  Ⓒ    │
                 └──────────────────┘
```

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 regs 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() →

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>