



# Advanced Java

*Trainer: Nilesh Ghule*



# Transactions

## MySQL

START TRANSACTION;

stmt1;

stmt2;

COMMIT;

or

ROLLBACK;

## JDBC

→ con.setAutoCommit(false);  
execute multiple sql statements  
e.g. stmt1.executeUpdate()  
stmt2.executeUpdate()

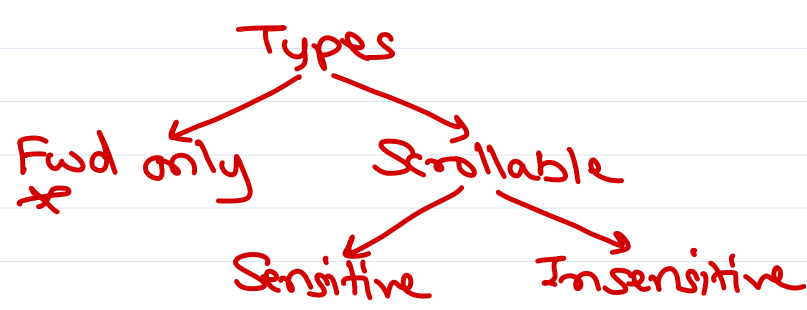
→ con.commit();

→ con.rollback();

Note: Spring @Transactional internally calls above jdbc methods to manage the tx automatically.

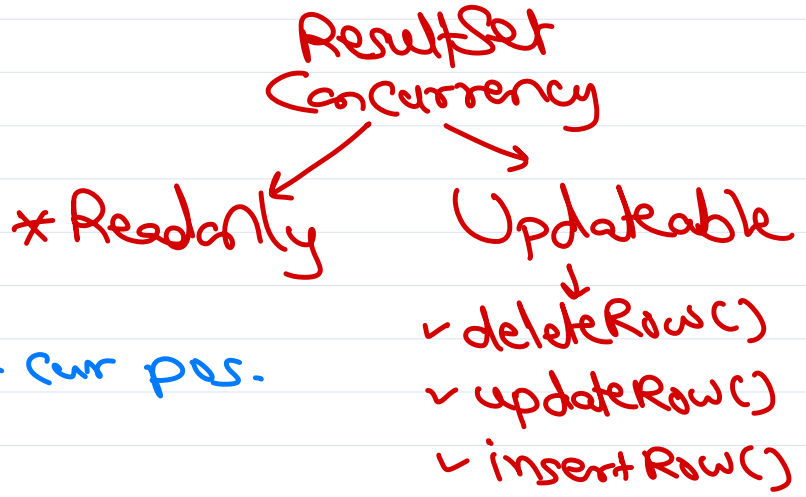
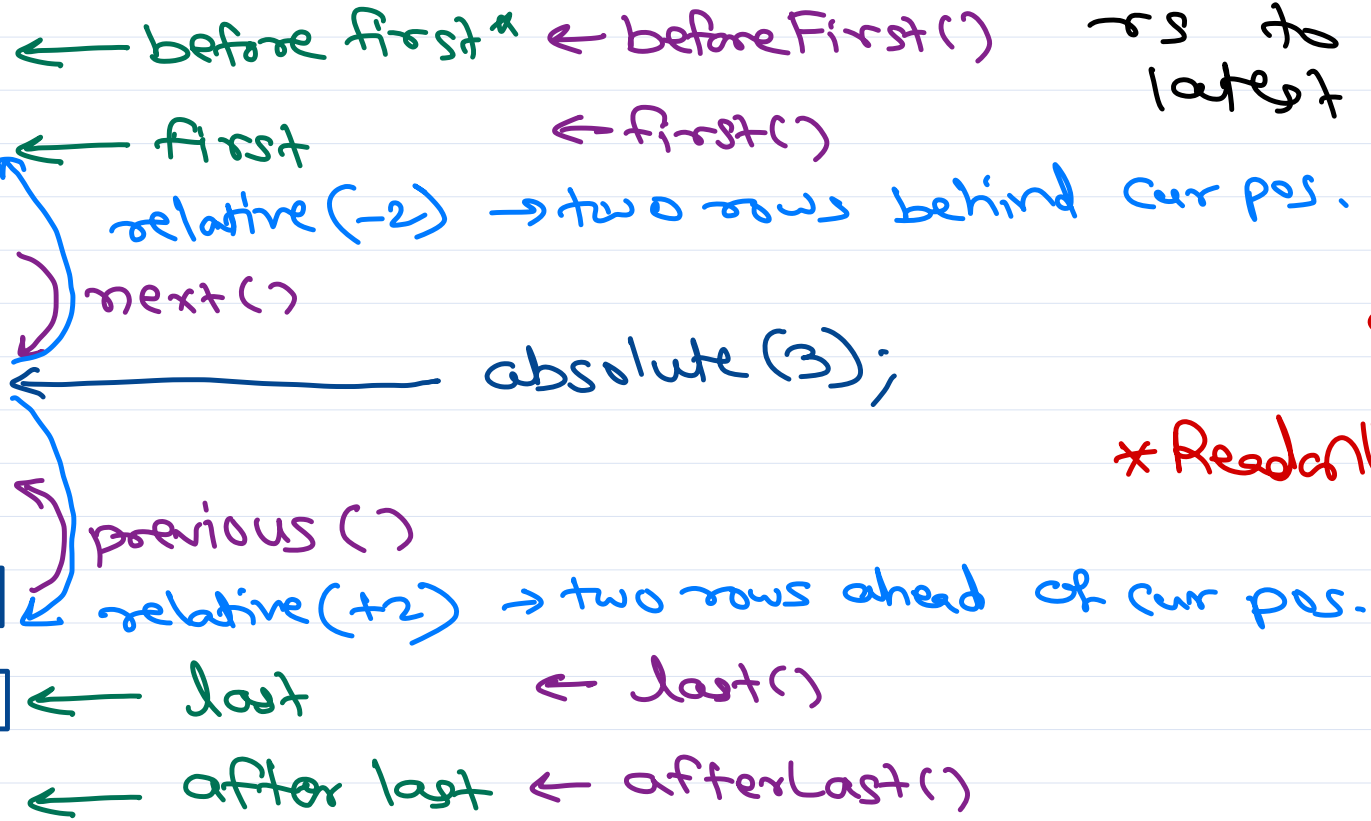
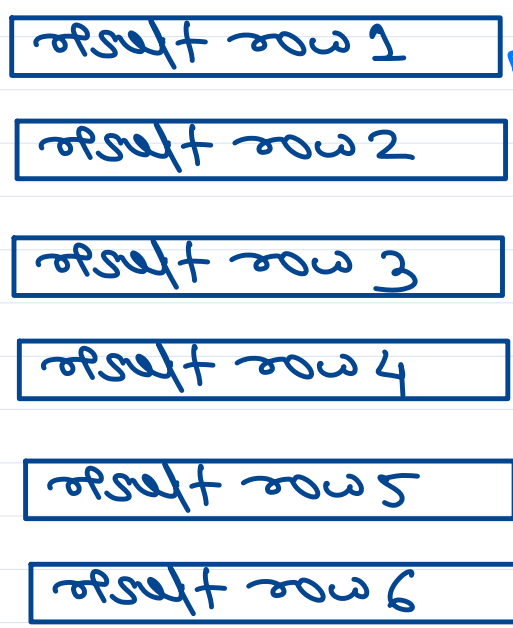


# ResultSet



**sensitive rs** → while rs is in process, if any changes done in db, they will be immediately available to rs.

**insensitive rs** → need to close & create new rs to get the latest changes from db.



# HTTP protocol

- ① server → program that provides service(s).
- ② client → program that consume service(s).

③ web server → program that allows to run one/more web appln in it.  
e.g. apache (php), tomcat (java), iis (.net/asp), jboss (java), ...  
\* Java web server → allows to run one/more java web appln.

④ web client → program that access web appln.  
e.g. browsers, mobile apps, ...

⑤ web appln → made up of multiple web pages.

web pages → static page or dynamic page

✓ static page → HTML pages (served to client as it is).

✓ dynamic page → Executed on server & produces HTML/XML/JSON output that is sent to client.

\* HTTP protocol → defines interaction betn web client & web server.

① request response model. → Req sent by client & then only resp generated & sent by server.

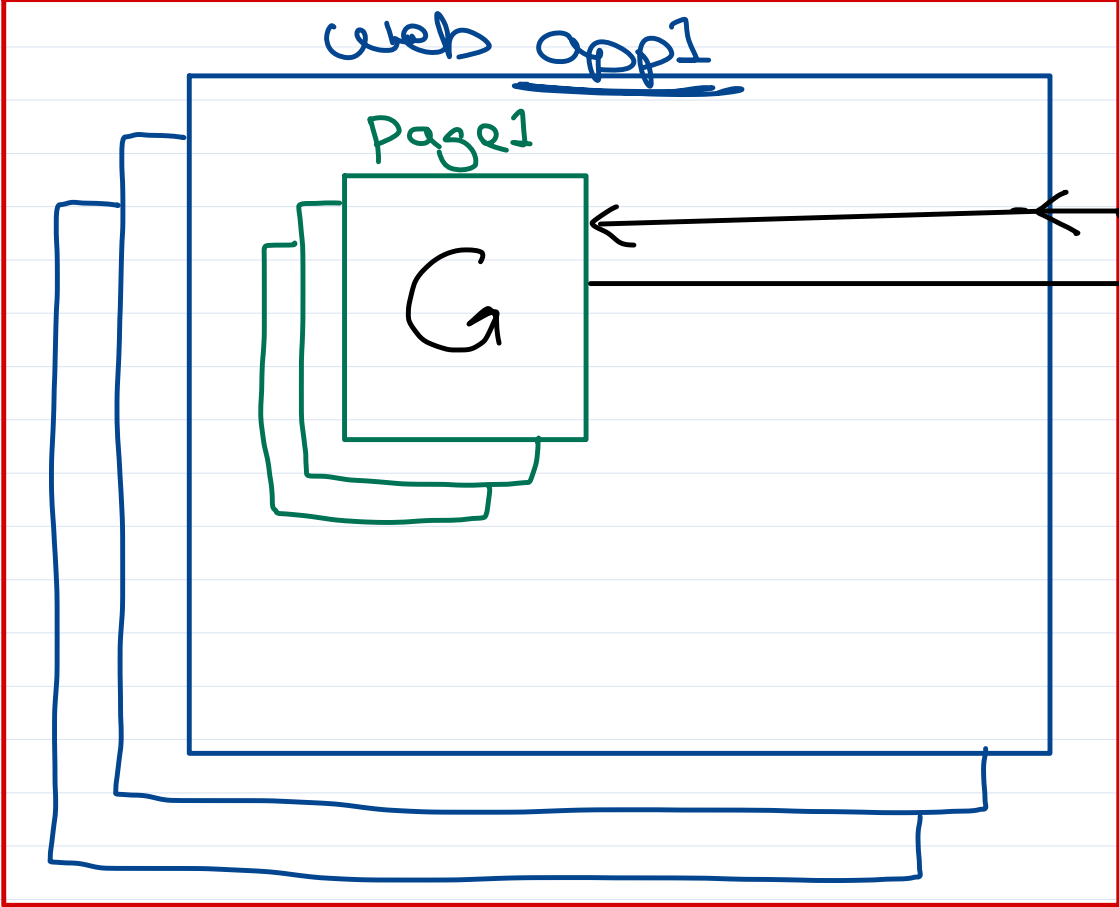
② Connection less protocol. → for each req new Conn may be established by client & closed when resp is sent.

③ stateless protocol. → by default, server do not remember any info/state of clients.



# HTTP protocol

web server (machine:port) http://machine:port/app1/page1



Resp Status  
1xx →  
2xx → Success  
3xx → redirection  
4xx → Client error  
5xx → Server error.

Req

web client



Resp

\* Content Types  
text/\*  
image/\*  
audio/\*  
video/\*  
application/\*

header:  
\* status: 200  
\* Content-length  
\* Content-type  
\* server info (ip...)  
\* Cookie

body: ...

Req

header:  
\* Server: port  
\* uri (/app1/page1)  
\* Content type  
\* Content Length  
\* User-agent: browser  
\* method: ... info  
\* client info (ip,...)  
\* Cookies  
...

body: ...

Req methods: ① GET ② POST ③ HEAD  
④ PUT ⑤ DELETE ⑥ TRACE ⑦ OPTIONS

⑧ PATCH



# HTTP Protocol

## Request Methods

- ① GET : gets a page from server.
  - data is sent (if any) in url (no body).
- ② POST : posts data (in form) to server.
  - data is sent in body
- ③ HEAD : Like GET but get only resp header
  - to get info about page/resource.
- ④ PUT : Put a resource on server
- ⑤ DELETE : Delete a resource from server.
- ⑥ TRACE : Used in debug issue
  - send req back as resp (loop back)
- ⑦ OPTIONS : Check which req methods are supported by resource/page.



# Servlet

\* Servlet is a Java class that is executed on server when req is received from client & produces resp that is sent to client.

\* One of the way to 'implement dynamic pages' in Java web appln.

```
@WebServlet("/hello")
public class MyServlet extends HttpServlet {
    @Override
    public void doGet(HttpServletRequest req,
        HttpServletResponse resp) throws ServletException, IOException {
        resp.setContentType("text/html");
        PrintWriter out = resp.getWriter();
        out.println("<html>");
        out.println("<body>");
        out.println("<h1> Hello, Dmc! </h1>");
        out.println(new Date());
        out.println("</body>");
        out.println("</html>");
    }
}
```

Servlet is a specs given in form of interfaces & classes.



*Thank you!*

Nilesh Ghule <nilesh@sunbeaminfo.com>

