**Create Web Bank Webpage...**

----------------------------------------------------------------------------------------------------

**Software We will requires are,**

1-> For reference sir's folder (https://drive.google.com/file/d/1RIL5SMc2A13sDRLtuUhXQMg9rq\_msbvc/view)

2-> Apache Tomcat (https://tomcat.apache.org/download-90.cgi)(download 32/64 bit windows service installer)

3-> MySQL 8.0 (https://dev.mysql.com/downloads/mysql/)\*if you want to download full version then select Windows (x86, 64-bit), ZIP Archive

Debug Binaries & Test Suite version or else just download Windows (x86, 64-bit), ZIP Archive.

4-> Notepad++ (https://notepad-plus-plus.org/downloads/v7.8.8/).

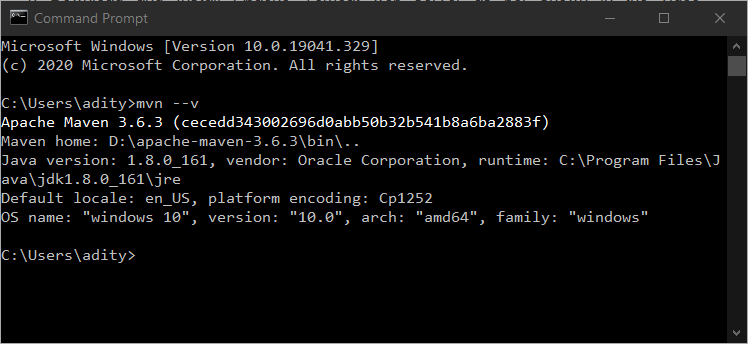
5-> MAVEN (https://maven.apache.org/download.cgi)

----------------------------------------------------------------------------------------------------

**STEPS TO BE FOLLOW TO CREATE A WEBSITE**

----------------------------------------------------------------------------------------------------

1. Download and Install Apache Tomcat web server as per shown in the video (<https://www.youtube.com/watch?v=OlOEYdLlPks>).
2. Download and Install MySQL 8.0 as per shown in the video (<https://www.youtube.com/watch?v=SYBIcjc1y0c>).
3. Download and Install notepad++.
4. Install Maven:
   1. Download Zip file.
   2. Extract the file in any of drive.
   3. Open the folder and click on bin folder.
   4. Copy MAVEN bin directory in START > CONTROL PANEL > ADMINISTRATIVE TOOL > SYSTEM INFORMATION > ENVIRONMENT VARIABLE ……… (or just simply click on Start and search for ENVIRONMENT VARIABLE)
   5. To Check Maven is properly installed open COMMAND PROMPT (click START and Search for command prompt)
   6. Type …. C:\Users\adity>mvn –version
   7. If this type of message is shown then your MAVEN is properly installed



1. Now create Folder named Mybankwebsite. (If you have knowledge how to create website using html/css)Then create website using .html (or just copy paste sir’s mybankwebsite whole folder.).
2. Create a new folder mybankjerseyservice. There will be four folder inside this folder 1.accounts 2.customer 3.eventsncy 4.moneytransfer
3. Open mybankjerseyservice folder in command Prompt and enter following commads.
   1. First we will create customer folder so enter this command-

mvn archetype:generate -DarchetypeArtifactId=jersey-quickstart-grizzly2 -DarchetypeGroupId=org.glassfish.jersey.archetypes -DinteractiveMode=false -DgroupId=mybank **-DartifactId=customer** -Dpackage=mybank -DarchetypeVersion=2.17

(Note:- -DartifactId= (“will be the name of your jersey service folder name”)….for customer folder artifact will be Dartifact=customer and for account folder artifact will be dartifact = account…and so on…)

1. Now one customer folder will be created on your desktop in that folder you will see 2 more folder one will be src and other will be pom.xml file.
2. Lets open that pom.xml file and make this following changes…
3. Note: words highlighted with in red are to give you information about that code
4. <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5. xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd">
6. <modelVersion>4.0.0</modelVersion>
7. <groupId>mybank</groupId> (Your sub folder name)
8. <artifactId>customer</artifactId> (Your folder name)
9. <packaging>jar</packaging>
10. <version>1.0-SNAPSHOT</version>
11. <name>customer</name>
12. <dependencyManagement>
13. <dependencies>
14. <dependency>
15. <groupId>org.glassfish.jersey</groupId>
16. <artifactId>jersey-bom</artifactId>
17. <version>${jersey.version}</version>
18. <type>pom</type>
19. <scope>import</scope>
20. </dependency>
21. </dependencies>
22. </dependencyManagement>
23. <dependencies>
24. <dependency>
25. <groupId>org.glassfish.jersey.containers</groupId>
26. <artifactId>jersey-container-grizzly2-http</artifactId>
27. </dependency>
28. <!-- uncomment this to get JSON support:
29. <dependency>
30. <groupId>org.glassfish.jersey.media</groupId>
31. <artifactId>jersey-media-moxy</artifactId>
32. </dependency>
33. -->
34. <dependency>
35. <groupId>junit</groupId>
36. <artifactId>junit</artifactId>
37. <version>4.9</version>
38. <scope>test</scope>
39. </dependency>
40. <dependency>
41. <groupId>mysql</groupId>
42. <artifactId>mysql-connector-java</artifactId> (Helps to connet mysql)
43. <version>8.0.18</version>
44. </dependency>
46. <!-- https://mvnrepository.com/artifact/com.googlecode.json-simple/json-simple -->
47. <dependency>
48. <groupId>com.googlecode.json-simple</groupId>
49. <artifactId>json-simple</artifactId> (helps to connect json files)
50. <version>1.1.1</version>
51. </dependency>
53. <dependency>
54. <groupId>mysql</groupId>
55. <artifactId>jdatabase</artifactId> (helps to connect database files)
56. <version>1.0</version>
57. </dependency>
59. <dependency>
60. <groupId>myevent</groupId>
61. <artifactId>jevent</artifactId> (helps to connect myevent folders)
62. <version>1.0</version>
63. </dependency>
64. </dependencies>
65. <build>
66. <plugins>
67. <plugin>
68. <groupId>org.apache.maven.plugins</groupId>
69. <artifactId>maven-compiler-plugin</artifactId>
70. <version>2.5.1</version>
71. <inherited>true</inherited>
72. <configuration>
73. <source>1.8</source> (Just change from 1.7 to 1.8)
74. <target>1.8</target> (Just change from 1.7 to 1.8)
75. </configuration>
76. </plugin>
77. <plugin>
78. <groupId>org.codehaus.mojo</groupId>
79. <artifactId>exec-maven-plugin</artifactId>
80. <version>1.2.1</version>
81. <executions>
82. <execution>
83. <goals>
84. <goal>java</goal>
85. </goals>
86. </execution>
87. </executions>
88. <configuration>
89. <mainClass>mybank.mainCustomer</mainClass> (mybamk is your subfolder in customer and mainCustomer is your java file in which your code works.)
90. </configuration>
91. </plugin>
92. </plugins>
93. </build>
94. <properties>
95. <jersey.version>2.17</jersey.version>
96. <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
97. </properties>
98. </project>
99. Now open src folder > main > java > mybank> main.java folder (Rename main.java to mainCustomer as we have mentioned in pom file). And now make following changes. Note: words highlighted with in red are to give you information about that code
100. package mybank;
101. import org.glassfish.grizzly.http.server.HttpServer;
102. import org.glassfish.jersey.grizzly2.httpserver.GrizzlyHttpServerFactory;
103. import org.glassfish.jersey.server.ResourceConfig;
104. import org.glassfish.grizzly.http.server.StaticHttpHandler;
105. import java.io.IOException;
106. import java.net.URI;
107. /\*\*
108. \* Main class.
109. \*
110. \*/
111. public class mainCustomer {
112. // Base URI the Grizzly HTTP server will listen on
113. public static final String BASE\_URI = "http://localhost:8081/myapp/";
114. (your Customer program will run on localhost:8081 (or port number which you will give))
115. /\*\*
116. \* Starts Grizzly HTTP server exposing JAX-RS resources defined in this application.
117. \* @return Grizzly HTTP server.
118. \*/
119. public static HttpServer startServer() {
120. // create a resource config that scans for JAX-RS resources and providers
121. // in mybank package
122. final ResourceConfig rc = new ResourceConfig().packages("mybank");
123. // create and start a new instance of grizzly http server
124. // exposing the Jersey application at BASE\_URI
125. return GrizzlyHttpServerFactory.createHttpServer(URI.create(BASE\_URI), rc);
126. }
127. /\*\*
128. \* Main method.
129. \* @param args
130. \* @throws IOException
131. \*/
132. public static void main(String[] args) throws IOException {
133. final HttpServer server = startServer();
134. server.getServerConfiguration().addHttpHandler(new StaticHttpHandler("D:\\Projects\\Laptop\_Project\\MyBank\\mybankwebsite\\"),"/");
135. (Path will be your website path.. this path is mine)
137. System.out.println(String.format("Jersey app started with WADL available at "
138. + "%sapplication.wadl\nHit enter to stop it...", BASE\_URI));
139. System.in.read();
140. server.stop();
141. }
142. }
143. Now open src folder > main > java > mybank> myResource.java folder (Rename myResources.java to svcCustomer as we have mentioned in mainCustomer.java file). Code will be as following…
144. package mybank;
145. import javax.ws.rs.GET;
146. import javax.ws.rs.FormParam;
147. import javax.ws.rs.POST;
148. import javax.ws.rs.Path;
149. import javax.ws.rs.Produces;
150. import javax.ws.rs.Consumes;
151. import javax.ws.rs.core.MediaType;
152. import javax.xml.ws.http.HTTPException;
153. import javax.ws.rs.PathParam;
154. import javax.ws.rs.QueryParam;
155. import javax.ws.rs.core.Response;
156. import org.w3c.dom.Document;
157. import org.w3c.dom.Element;
158. //simple JSON
159. import org.json.simple.JSONArray;
160. import org.json.simple.JSONObject;
161. import org.json.simple.parser.\*;
162. //common bank database object
163. import jdatabase.\*; (It will import files from jdatabase folder)
164. import jevent.\*; (It will import files from jevent folder)
165. /\*\*
166. \* Root resource (exposed at "svcCustomer" path)
167. \*/
168. @Path("customer")
169. public class svcCustomer {
171. public String data;
172. public dbsql db;
173. public String query;
174. public jeventClient eventclient;
175. final int READ\_QUERY = 0;
176. final int WRITE\_QUERY = 1;
177. public svcCustomer(){
178. db = new dbsql(0); //create the database object
179. eventclient = new jeventClient("customer");
180. }
182. @POST
183. @Path("/executequery")
184. @Produces({"application/json"})
185. @Consumes({"application/json"})
186. public Response executequery(String data) {
188. String jsonresult="";
190. System.out.println("svcCustomer executequery called");
192. try{
194. data = data.replace("\n", "").replace("\\r", "").replace("\t", "");

197. // parsing file "JSONExample.json"
198. Object obj = new JSONParser().parse(data);
200. // typecasting obj to JSONObject
201. JSONObject jo = (JSONObject) obj;
202. //get the query and query type
203. String query = (String) jo.get("query");
204. int querytype = ((Long)jo.get("querytype")).intValue();
205. jsonresult = db.executequery(query,querytype) ; //return json result from the query
207. System.out.println("svcCustomer executequery result=" + jsonresult);
209. //at present we are just forwarding the query and querytype as it is received
210. //we may have to create a standard json data for events which contains source, destination
211. eventclient.sendEvent(data);
212. }
213. catch(Exception e)
214. {
215. System.out.println("returning post2 failure");
216. System.out.println("[ {'error':'" + e.toString() + "'}]");
217. return sendjsonresponse("[ {'error':'" + e.toString() + "'}]"); //send the error as response
218. }
220. return sendjsonresponse(jsonresult);
221. }
223. public Response sendjsonresponse(String output)
224. {
225. try {
226. data = output;
227. return Response.status(200).entity(data).build();
228. }
229. catch(Exception e) {
230. throw new HTTPException(400);
231. }
232. }
234. @GET
235. @Produces(MediaType.TEXT\_PLAIN)
236. public String getIt() {
237. return "<h2>Welcome to MyBank Customer service</h2>";
238. }

241. public void sendEvent(String eventdata)
242. {
243. //connect to event service
244. //send event to event sync
245. eventclient.sendEvent(eventdata);
247. }
249. //sync the event locally as received from event synchronizer
250. public void syncEvent(String eventdata)
251. {
252. //connect to event service
253. //sync event into local database
254. eventclient.syncEvent(eventdata,db);
256. }
257. }
258. This finishes your Customer Folder.. in same way you have to create accounts, eventsync and moneytransfer service…. Codes will be remain same Instead of Customer use Account that’s it….. (I am providing zip folder of project you can refer it and do according to it….)
259. Now we will Create Database folder …for that create new folder and rename it with mybankdatabase.
260. Now open mybankdatabase folder and create two more folders in it as jdatabase (You can give any name but it was mentioned in sirs project you can use jdatabase for less confusion) and another named as sqlscriptdata….
261. To Run further you need to download and install mysql 8.0 in your PC. And there you have to give one user id and password further you will require it ….
262. First we will work on jdatabase folder so click on jdatabase folder and create a new file and type this code and save it as dbsql.java
263. package jdatabase;
265. import java.sql.\*;
267. //simple JSON
268. import org.json.simple.JSONArray;
269. import org.json.simple.JSONObject;
270. import org.json.simple.parser.\*;
272. public class dbsql
273. {
274. final static int READ\_QUERY = 0;
275. final static int WRITE\_QUERY = 1;
277. static String databasename = "dbCustomer"; (Your customer json file will call out…)
278. static String databaseusername = "root"; (Your mysql 8.0 userid)
279. static String databasepassword = "aditya"; (Your mysql 8.0 pasword)
280. static String databasetestquery = "SELECT CONCAT( " +
281. " '[', " +
282. " GROUP\_CONCAT(JSON\_OBJECT('\_id', \_id)), " +
283. "']'" +
284. ") as jsonresult from tCustomer;";
286. static serviceCommon cmn = new serviceCommon();

289. //can test the sql independently from command prompt too
290. public static void main(String args[]){
291. try
292. {
293. String data ;
294. String query = databasetestquery;
296. initializedatabase(0);
298. //data = executeQuery("SELECT \* FROM \_qbinv",READ\_QUERY);
299. System.out.println("Executing query :" + query);
300. data = executequery(query,READ\_QUERY);
302. System.out.println(data);
303. }
304. catch(Exception e)
305. {
306. System.out.println(e.toString());
307. }
308. }
310. /\* constructor \*/
311. /\*
312. databaseID = 0 dbCustomer, 1 dbAccount , 2 dbMoneyTransfer, 3 dbEventSynch
313. \*/
314. public dbsql(int databaseID)
315. {
316. initializedatabase(databaseID); //bydefault connect to dbCustomer
317. }
319. public static void initializedatabase(int databaseID)
320. {
321. try
322. {
323. String dbConfigFileName = "dbconfig\_customer.json";
325. System.out.println("databaseid=" + databaseID);
327. if (databaseID == 0)
328. dbConfigFileName = "dbconfig\_customer.json";
329. else if (databaseID == 1)
330. dbConfigFileName = "dbconfig\_account.json";
331. else if (databaseID == 2)
332. dbConfigFileName = "dbconfig\_moneytransfer.json";
333. else if (databaseID == 3)
334. dbConfigFileName = "dbconfig\_eventsync.json";
335. System.out.println("reading configuration file" + dbConfigFileName);
337. databasename = cmn.getJSONStringValuefromFile(dbConfigFileName,  "databasename");
338. databaseusername = cmn.getJSONStringValuefromFile(dbConfigFileName,  "databaseusername");
339. databasepassword = cmn.getJSONStringValuefromFile(dbConfigFileName,  "databasepassword");
340. databasetestquery = cmn.getJSONStringValuefromFile(dbConfigFileName,  "databasetestquery");
342. System.out.println("initializing database");
344. System.out.println("databasename="+databasename);
345. System.out.println("databaseusername="+databaseusername);
346. System.out.println("databasepassword="+databasepassword);
347. System.out.println("databasetestquery="+databasetestquery);
348. }
349. catch(Exception e){
350. System.out.println("jdatabase error :" + e.toString());
351. }
353. }
355. /\* querytype = 0 means SELECT query, querytype 1 means data manipulation query \*/
356. // create the mysql insert preparedstatement
357. /\*
358. //EXAMPLE OF DATA MANIPULATION QUERY
359. PreparedStatement preparedStmt = conn.prepareStatement(query);
360. preparedStmt.setString (1, "Barney");
361. preparedStmt.setString (2, "Rubble");
362. preparedStmt.setDate   (3, startDate);
363. preparedStmt.setBoolean(4, false);
364. preparedStmt.setInt    (5, 5000);
365. // execute the preparedstatement
366. preparedStmt.execute();
367. \*/
369. public static String executequery(String query, int querytype)
370. {
372. Statement stmt; //used for SELECT query
373. PreparedStatement preparedStmt; //used for UPDATE, DELETE, INSERT
374. ResultSet rs;
375. String sResult;
377. sResult = "";
378. boolean displayResult = true;
379. String querydelimiter = ":::";
381. try
382. {
383. System.out.println("attempting query against database:"+databasename+";user="+databaseusername+";password="+databasepassword);
385. Class.forName("com.mysql.cj.jdbc.Driver");
386. Connection con=DriverManager.getConnection(
387. "jdbc:mysql://localhost:3306/" + databasename,databaseusername,databasepassword);
389. stmt=con.createStatement();
391. if (querytype == READ\_QUERY)
392. {
394. displayResult = false;
395. rs=stmt.executeQuery(query);
396. while(rs.next())
397. {
398. sResult = rs.getString(1);
399. }
401. }
402. else if (querytype == WRITE\_QUERY)
403. {
404. System.out.println("executing query type 1 : " + query );
406. String[] querylist = query.split(querydelimiter);
407. String q = "";
408. System.out.println("Total queries to execute = " + querylist.length);
410. for (int i = 0; i < querylist.length; i++) {
411. q = querylist[i]; //+ ";"; //add the semi colon back for each query
412. System.out.print("\n\n executing query=" + q);
413. preparedStmt = con.prepareStatement(q);
414. preparedStmt.execute();
415. }
417. sResult = "[{'success': 'WRITE\_QUERY executed'}]";
418. }
420. con.close();
422. }
423. catch(Exception e)
424. {
425. sResult = "[{'error':'" + e.toString() +"'}]";
426. }
428. if (displayResult)
429. System.out.println("Result: " + sResult);
430. else
431. System.out.println("successful read query");
433. return sResult;
434. }
435. }
436. And then after that create another file and type this code and save it as serviceCommon.java. We will later compile it both file using **bat** file it is an shortcut to compile whole folder together…
437. package jdatabase;
438. //simple JSON
439. import org.json.simple.JSONArray;
440. import org.json.simple.JSONObject;
441. import org.json.simple.parser.\*;
442. import java.io.FileNotFoundException;
443. import java.io.FileReader;
444. import java.io.IOException;
445. import java.io.FileWriter;
446. /\* class to have all common functions \*/
447. public class serviceCommon {
448. public static void main(String[] args) throws IOException {
450. System.out.println("Reading JSON ..");
451. System.out.println("websitefolder=" + getJSONStringValuefromFile("dbconfig.json","websitefolder"));
453. }
454. public static String getJSONStringValuefromFile(String filename, String node)
455. {
456. String nodevalue = "";
458. try (FileReader config = new FileReader(filename))
459. {
460. Object obj = new JSONParser().parse(config);
462. // typecasting obj to JSONObject
463. JSONObject jo = (JSONObject) obj;
465. // getting firstName and lastName
466. nodevalue = (String) jo.get(node);
467. }
468. catch (FileNotFoundException e) {
469. e.printStackTrace();
470. } catch (IOException e) {
471. e.printStackTrace();
472. } catch (ParseException e) {
473. e.printStackTrace();
474. }
476. return nodevalue;
477. }
478. }
479. Here we finish jdabase folder coding .
480. now get back to sqlscriptdat folder for that either you can code on notepad ++ or code using mysql 8.0 but mysql 8.0 will do better… So open Mysql8.0 workbench and connect to instance 3306 enter your username and user password…and then you will get a query tab in that type the following code and save it as **create\_customer\_database** in sqlscripdata location
481. DROP DATABASE IF EXISTS dbCustomer;
482. CREATE DATABASE IF NOT EXISTS dbCustomer;
483. USE dbCustomer;
484. SET GLOBAL group\_concat\_max\_len = 1000000;
485. /\*(account type for each of the accounts in chart of accounts)\*/
486. CREATE TABLE tCustomer (
487. \_id INT NOT NULL AUTO\_INCREMENT,
488. customername VARCHAR(255),
489. customeraddress VARCHAR(255),
490. customerphone VARCHAR(255),
491. customertype VARCHAR(255),
492. createdate timestamp default now(),
493. lastmodifieddate timestamp default now(),
494. createdby VARCHAR(255) default 'SYS',
495. lastmodifiedby VARCHAR(255) default 'SYS',
496. PRIMARY KEY (\_id)
497. );
498. LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\tCustomer.csv'
499. INTO TABLE tCustomer
500. FIELDS TERMINATED BY ','
501. ENCLOSED BY '"'
502. LINES TERMINATED BY '\n'
503. (
504. customername ,
505. customeraddress ,
506. customerphone ,
507. customertype
508. );
509. /\* event table \*/
510. /\* microservices have a common event receiving tables \*/
511. /\*
512. eventdirection - -1 does not mean anything, 0 means its event waiting to be sent, 1 means event is sent, 2 means event is received
513. all received events are updated with direction of 2, all events ready to be sent are marked with a direction of 0
514. eventstatus - 0 means event is not processed as yet, 1 means event is processed
515. event source - which service has created this event (0 means customer, 1 means account, 2 means money transfer)
516. event destination - which service is suppose to receive the event (0 means customer, 1 means account, 2 means money transfer)
517. \*/
518. CREATE TABLE tEvents (
519. \_id INT NOT NULL AUTO\_INCREMENT,
520. eventid INT NOT NULL ,
521. eventsource VARCHAR(400) default "",
522. eventdestination VARCHAR(400) default "",
523. eventdata json DEFAULT NULL,
524. eventstatus INT default 0 ,
525. eventdirection INT default -1,
526. createdate timestamp default now(),
527. lastmodifieddate timestamp default now(),
528. createdby VARCHAR(255) default 'SYS',
529. lastmodifiedby VARCHAR(255) default 'SYS',
530. PRIMARY KEY (\_id)
531. );
532. In same way type code for account, event sync and moneytransfer.
     1. **create\_account\_database (code)**
533. DROP DATABASE IF EXISTS dbAccount;
534. CREATE DATABASE IF NOT EXISTS dbAccount;
535. USE dbAccount;
536. SET GLOBAL group\_concat\_max\_len = 1000000;
537. /\*(account type for each of the accounts in chart of accounts)\*/
538. CREATE TABLE tAccount (
539. \_id INT NOT NULL AUTO\_INCREMENT,
540. accountname VARCHAR(255),
541. accountbalance VARCHAR(255),
542. createdate timestamp default now(),
543. lastmodifieddate timestamp default now(),
544. createdby VARCHAR(255) default 'SYS',
545. lastmodifiedby VARCHAR(255) default 'SYS',
546. PRIMARY KEY (\_id)
547. );
548. LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\tAccount.csv'
549. INTO TABLE tAccount
550. FIELDS TERMINATED BY ','
551. ENCLOSED BY '"'
552. LINES TERMINATED BY '\n'
553. (
554. accountname ,
555. accountbalance
556. );
557. /\* event table \*/
558. /\* microservices have a common event receiving tables \*/
559. /\*
560. eventdirection - -1 does not mean anything, 0 means its event waiting to be sent, 1 means event is sent, 2 means event is received
561. eventstatus - 0 means event is not processed as yet, 1 means event is processed
562. event source - which service has created this event (0 means customer, 1 means account, 2 means money transfer)
563. event destination - which service is suppose to receive the event (0 means customer, 1 means account, 2 means money transfer)
565. \*/
566. /\* event table \*/
567. /\* microservices have a common event receiving tables \*/
568. /\*
569. eventdirection - -1 does not mean anything, 0 means its event waiting to be sent, 1 means event is sent, 2 means event is received
570. all received events are updated with direction of 2, all events ready to be sent are marked with a direction of 0
571. eventstatus - 0 means event is not processed as yet, 1 means event is processed
572. event source - which service has created this event (0 means customer, 1 means account, 2 means money transfer)
573. event destination - which service is suppose to receive the event (0 means customer, 1 means account, 2 means money transfer)
574. \*/
575. CREATE TABLE tEvents (
576. \_id INT NOT NULL AUTO\_INCREMENT,
577. eventid INT NOT NULL ,
578. eventsource VARCHAR(400) default "",
579. eventdestination VARCHAR(400) default "",
580. eventdata json default NULL,
581. eventstatus INT default 0 ,
582. eventdirection INT default -1,
583. createdate timestamp default now(),
584. lastmodifieddate timestamp default now(),
585. createdby VARCHAR(255) default 'SYS',
586. lastmodifiedby VARCHAR(255) default 'SYS',
587. PRIMARY KEY (\_id)
588. );
     1. **create\_eventsync\_database (code)**
589. DROP DATABASE IF EXISTS dbEventSync;
590. CREATE DATABASE IF NOT EXISTS dbEventSync;
591. USE dbEventSync;
592. SET GLOBAL group\_concat\_max\_len = 1000000;
593. /\* event table \*/
594. /\* microservices have a common event receiving tables \*/
595. /\*
596. eventdirection - -1 does not mean anything, 0 means its event waiting to be sent, 1 means event is sent, 2 means event is received
597. all received events are updated with direction of 2, all events ready to be sent are marked with a direction of 0
598. eventstatus - 0 means event is not processed as yet, 1 means event is processed
599. event source - which service has created this event (0 means customer, 1 means account, 2 means money transfer)
600. event destination - which service is suppose to receive the event (0 means customer, 1 means account, 2 means money transfer)
602. \*/
603. CREATE TABLE tEvents (
604. \_id INT NOT NULL AUTO\_INCREMENT,
605. eventid INT NOT NULL ,
606. eventsource VARCHAR(400) default "",
607. eventdestination VARCHAR(400) default "",
608. eventdata json DEFAULT NULL,
609. eventstatus INT default 0 ,
610. eventdirection INT default -1,
611. createdate timestamp default now(),
612. lastmodifieddate timestamp default now(),
613. createdby VARCHAR(255) default 'SYS',
614. lastmodifiedby VARCHAR(255) default 'SYS',
615. PRIMARY KEY (\_id)
616. );
617. LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL Server 8.0\\Uploads\\tEventSync.csv'
618. INTO TABLE tEvents
619. FIELDS TERMINATED BY ','
620. ENCLOSED BY '"'
621. LINES TERMINATED BY '\n'
622. (
623. eventdata,
624. eventstatus
625. );
627. Now you have to create csv file for that open excel sheet and give entry of accounts and save that file as tCustomer.csv, tAccount.csv, tEventSync.csv ….
628. Next copy those three csv files and open C drive search ProgramData in that search mysql > Mysql server 8.0> Upload > and save it there…
629. Now open again mybankdatabase folder and create new two json files
     1. dbconfig\_account.json
630. {
631. "softwarefolder":"D:\\Projects\\Laptop\_Project\\MyBank\\", (enter your website path )
632. "websitefolder":"D:\\Projects\\Laptop\_Project\\MyBank\\mybankwebsite\\", (enter your website path )
633. "databasename":"dbAccount",
634. "databaseusername":"root", (enter your mysql username )
635. "databasepassword":"aditya" , (enter your mysql password )
636. "databasetestquery":"SELECT CONCAT(  '[',  GROUP\_CONCAT(JSON\_OBJECT('\_id', \_id)), ']') as jsonresult from tAccount;"
637. }
     1. Dbconfig\_customer.json (do similar changes as above)
638. {
639. "softwarefolder":"D:\\Projects\\Laptop\_Project\\MyBank\\",
640. "websitefolder":"D:\\Projects\\Laptop\_Project\\MyBank\\mybankwebsite\\",
641. "databasename":"dbCustomer",
642. "databaseusername":"root",
643. "databasepassword":"aditya" ,
644. "databasetestquery":"SELECT CONCAT(  '[',  GROUP\_CONCAT(JSON\_OBJECT('\_id', \_id)), ']') as jsonresult from tAccount;"
645. }
646. Now to execute whole folder or to compile whole folder we will create bat file …in short without typing compiling code every time if we 2x click on that bat file it will automatically run whole folder and give you output …
647. Code will be as follows….
648. TITLE jdatabase (folder name)
649. echo off
650. cls
651. echo "COMPILING jdatabase code .. "
652. javac -cp "json-simple-1.1.1.jar";"mysql-connector-java-8.0.18.jar"; jdatabase\\*.java
653. echo "CREATING jdatabase jar"  (create jar folder of jdatabase similar to zip type )
654. jar cvf jdatabase.jar ./jdatabase/\*.class json-simple-1.1.1.jar mysql-connector-java-8.0.18.jar
655. echo "INSTALLING mysql-connector-java-8.0.18 jar files maven repository .. "
656. start "MVN INSTALLING mysql-connector-java-8.0.18 jar files maven repository" cmd.exe /k "mvn install:install-file -Dfile=D:\Projects\Laptop\_Project\MyBank\mybankdatabase\mysql-connector-java-8.0.18.jar -DgroupId=mysql -DartifactId=mysql-connector-java -Dversion=8.0.18 -Dpackaging=jar -DgeneratePom=true" (Path will be yours bankdatabase path)
657. echo "INSTALLING json-simple-1.1.1 jar files maven repository .. "
659. start "MVN INSTALLING json-simple-1.1.1 jar files maven repository .. " cmd.exe /k "mvn install:install-file -Dfile=D:\Projects\Laptop\_Project\MyBank\mybankdatabase\json-simple-1.1.1.jar -DgroupId=com.googlecode.json-simple -DartifactId=json-simple -Dversion=1.1.1 -Dpackaging=jar -DgeneratePom=true"
660. (Path will be yours bankdatabase path)
661. echo "INSTALLING jdatabase jar files maven repository .. "
662. start "MVN INSTALLING jdatabase jar files maven repository .. " cmd.exe /k "mvn install:install-file -Dfile=D:\Projects\Laptop\_Project\MyBank\mybankdatabase\jdatabase.jar -DgroupId=mysql -DartifactId=jdatabase -Dversion=1.0 -Dpackaging=jar -DgeneratePom=true"
663. (Path will be yours bankdatabase path)
664. echo "TESTING jdatabase connection .."
665. echo
666. java -cp "json-simple-1.1.1.jar";"mysql-connector-java-8.0.18.jar"; jdatabase.dbsql
667. It will create json-simpile—1.1.1.jar …. Jdatabase.jar …..mysqlconnector.jar …. mysql-connector-java-8.0.18.jar files in your jdatabase folder ….Here we complete jdatabase folder work….
668. Now we will create mybankeventclient folder for that run maven command …

mvn archetype:generate -DarchetypeArtifactId=jersey-quickstart-grizzly2 -DarchetypeGroupId=org.glassfish.jersey.archetypes -DinteractiveMode=false -DgroupId=jevent **-DartifactId=mybankeventclient** -Dpackage=jevent -DarchetypeVersion=2.17

1. Like previous customer folder it will create in similar manner …..Just make changes in pom.xml as follows
2. <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3. xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd">
4. <modelVersion>4.0.0</modelVersion>
5. <groupId>myevent</groupId>
6. <artifactId>jeventclient</artifactId>
7. <packaging>jar</packaging>
8. <version>1.0-SNAPSHOT</version>
9. <name>jeventclient</name>
10. <dependencyManagement>
11. <dependencies>
12. <dependency>
13. <groupId>org.glassfish.jersey</groupId>
14. <artifactId>jersey-bom</artifactId>
15. <version>${jersey.version}</version>
16. <type>pom</type>
17. <scope>import</scope>
18. </dependency>
19. </dependencies>
20. </dependencyManagement>
21. <dependencies>
22. <dependency>
23. <groupId>org.glassfish.jersey.containers</groupId>
24. <artifactId>jersey-container-grizzly2-http</artifactId>
25. </dependency>
26. <!-- uncomment this to get JSON support:
27. <dependency>
28. <groupId>org.glassfish.jersey.media</groupId>
29. <artifactId>jersey-media-moxy</artifactId>
30. </dependency>
31. -->
32. <dependency>
33. <groupId>junit</groupId>
34. <artifactId>junit</artifactId>
35. <version>4.9</version>
36. <scope>test</scope>
37. </dependency>
39. <dependency>
40. <groupId>mysql</groupId>
41. <artifactId>mysql-connector-java</artifactId>
42. <version>8.0.18</version>
43. </dependency>
45. <!-- https://mvnrepository.com/artifact/com.googlecode.json-simple/json-simple -->
46. <dependency>
47. <groupId>com.googlecode.json-simple</groupId>
48. <artifactId>json-simple</artifactId>
49. <version>1.1.1</version>
50. </dependency>
52. <dependency>
53. <groupId>mysql</groupId>
54. <artifactId>jdatabase</artifactId>
55. <version>1.0</version>
56. </dependency>
57. </dependencies>
58. <build>
59. <plugins>
60. <plugin>
61. <groupId>org.apache.maven.plugins</groupId>
62. <artifactId>maven-compiler-plugin</artifactId>
63. <version>2.5.1</version>
64. <inherited>true</inherited>
65. <configuration>
66. <source>1.7</source>
67. <target>1.7</target>
68. </configuration>
69. </plugin>
70. <plugin>
71. <groupId>org.codehaus.mojo</groupId>
72. <artifactId>exec-maven-plugin</artifactId>
73. <version>1.2.1</version>
74. <executions>
75. <execution>
76. <goals>
77. <goal>java</goal>
78. </goals>
79. </execution>
80. </executions>
81. <configuration>
82. <mainClass>jevent.jeventClient</mainClass>
83. </configuration>
84. </plugin>
85. </plugins>
86. </build>
87. <properties>
88. <jersey.version>2.17</jersey.version>
89. <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
90. </properties>
91. </project>
92. Now open src>main>java>jevent>now rename main.java as jeventclient.java and delete myResource file ….
93. package jevent;
94. import java.net.URI;
95. import javax.ws.rs.client.Client;
96. import javax.ws.rs.client.ClientBuilder;
97. import javax.ws.rs.client.WebTarget;
98. import javax.ws.rs.core.MediaType;
99. import javax.ws.rs.core.Response;
100. import javax.ws.rs.core.UriBuilder;
101. import javax.ws.rs.client.Entity;
102. import javax.ws.rs.core.Form;
103. import org.glassfish.jersey.client.ClientConfig;
104. //simple JSON
105. import org.json.simple.JSONArray;
106. import org.json.simple.JSONObject;
107. import org.json.simple.parser.\*;
108. import javax.xml.ws.http.HTTPException;
109. import jdatabase.\*;
110. public class jeventClient
111. {
112. private static String eventsyncURI = "http://localhost:8084/myapp";
113. private static WebTarget service = null;
114. private static String eventsource;
115. private static Client client;
117. public jeventClient(String eventsrc)
118. {
119. //initialize rest client to event sync
120. ClientConfig config = new ClientConfig();
121. client = ClientBuilder.newClient(config);
122. service  = client.target(getBaseURI(eventsyncURI));
123. eventsource = eventsrc;
124. }
126. //can test the sql independently from command prompt too
127. public static void main(String args[]){
128. System.out.println("event client is initialized sync events at " + eventsyncURI);
129. }
130. public Response sendevent(JSONObject inputJsonObj)
131. {
132. Response resp =
133. service.path("event")
134. .path("sync")
135. .request(MediaType.APPLICATION\_JSON)
136. .post(Entity.json(inputJsonObj.toString()));
138. String output = resp.readEntity(String.class);
139. System.out.println(output);
141. return resp;
142. }
143. public Response sendEvent(String data)
144. {
145. String output= "";
147. try
148. {
149. data = data.replace("\n", "").replace("\\r", "").replace("\t", "");
150. Object obj = new JSONParser().parse(data);
151. JSONObject inputJsonObj  = (JSONObject) obj;
153. Response resp =
154. service.path("event")
155. .path("sync")
156. .request(MediaType.APPLICATION\_JSON)
157. .post(Entity.json(inputJsonObj.toString()));
159. output = resp.readEntity(String.class);
160. System.out.println(output);
161. }
162. catch(Exception e )
163. {
164. System.out.println("returning sendevent failure");
165. System.out.println("[ {'error':'" + e.toString() + "'}]");
166. return sendjsonresponse("[ {'error':'" + e.toString() + "'}]"); //send the error as response
167. }
169. return sendjsonresponse(output);
170. }
172. // syncevents are raised by eventsync. It should be processed by microservices as needed
173. // the component seeking to sync the event also passes the database into which the event needs to be synced
174. public String syncEvent(String eventdata, dbsql db)
175. {
176. //store the event in local database as per dbConfigFileName
177. //return status 0, 1
178. String query,jsonresult;
179. int querytype = 1; //insert, update, delete queries are set to 1
181. // since we are synching locally eventdestination is the source that is set
182. // we do not know who sent the message,hence event source is unknown for now
183. // eventstatus = 0 means event is saved and needs to be processed
184. // eventdirection = 2 since its a received event, waiting to be synched locally
186. //json data needs the escape character to understand single quote contained within query
187. eventdata = eventdata.replace("'","\\'");
189. query = "INSERT INTO tEvents " +
190. " (eventdata, " +
191. " eventdestination, " +
192. " eventstatus, " +
193. " eventdirection, " +
194. " eventid, " +
195. " eventsource) " +
196. " VALUES ("  +
197. "'" + eventdata + "'" + "," +
198. "'" + eventsource + "'" + "," +
199. "0"  + "," +
200. "2"  + "," +
201. "-1"  + "," +
202. "'" + "unknown" + "'" +
203. ");";
205. System.out.println("query to be fired =" + query);
207. jsonresult = db.executequery(query,querytype) ; //return json result from the query
209. System.out.println("svcCustomer executequery result=" + jsonresult);
210. return jsonresult;
211. }
213. public Response broadcastEvent(String data, String uri)
214. {
215. String output= "";
217. try
218. {
220. System.out.println("broadcasting to service at :" + uri);
222. data = data.replace("\n", "").replace("\\r", "").replace("\t", "");
223. Object obj = new JSONParser().parse(data);
224. JSONObject inputJsonObj  = (JSONObject) obj;
226. WebTarget broadcasttoservice  = client.target(uri);
228. Response resp =
229. broadcasttoservice
230. .request(MediaType.APPLICATION\_JSON)
231. .post(Entity.json(inputJsonObj.toString()));
233. output = resp.readEntity(String.class);
234. System.out.println(output);
235. }
236. catch(Exception e )
237. {
238. System.out.println("returning broadcast event failure");
239. System.out.println("[ {'error':'" + e.toString() + "'}]");
240. return sendjsonresponse("[ {'error':'" + e.toString() + "'}]"); //send the error as response
241. }
243. return sendjsonresponse(output);
244. }
246. private static URI getBaseURI(String uri) {
247. return UriBuilder.fromUri(uri).build();
248. }
250. public Response sendjsonresponse(String output)
251. {
252. String data = "";
253. try {
254. data = output;
255. return Response.status(200).entity(data).build();
256. }
257. catch(Exception e) {
258. throw new HTTPException(400);
259. }
260. }
261. } // eventClient class end
262. Now again go inside mybankeventclient folder and add dbconfig\_eventclient.json file
263. {
264. "softwarefolder":"D:\\Projects\\Laptop\_Project\\MyBank\\",
265. "websitefolder":"D:\\Projects\\Laptop\_Project\\MyBank\\mybankwebsite\\",
266. "databasename":"dbAccount",
267. "databaseusername":"root",
268. "databasepassword":"aditya",
269. "databasetestquery":"SELECT CONCAT(  '[',  GROUP\_CONCAT(JSON\_OBJECT('\_id', \_id)), ']') as jsonresult from tAccount;"
270. }
271. Like before we have created bat file for mybankdatabase folder in same way we have to create bat file for mybankeventclient open notepad++ and save it as testjeventClient.bat Make changes in Path put your file location
272. title jeventClient
273. echo off
274. cls
275. echo "COMPILING"
276. start "MVN COMPILE JEVENTCLIENT" cmd.exe /k "mvn clean compile"
277. timeout 10 > NUL
278. echo "CREATING jeventClient jar"
279. cd .\target\classes
280. jar cvf jevent.jar jevent\\*.class
281. cd ..
282. cd ..
283. copy .\target\classes\jevent.jar
284. echo "INSTALLING mysql-connector-java-8.0.18 jar files maven repository .. "
285. start "MVN INSTALLING mysql-connector-java-8.0.18 jar files maven repository" cmd.exe /k "mvn install:install-file -Dfile=D:\Projects\Laptop\_Project\MyBank\mybankdatabase\mysql-connector-java-8.0.18.jar -DgroupId=mysql -DartifactId=mysql-connector-java -Dversion=8.0.18 -Dpackaging=jar -DgeneratePom=true"
286. echo "INSTALLING json-simple-1.1.1 jar files maven repository .. "
288. start "MVN INSTALLING json-simple-1.1.1 jar files maven repository .. " cmd.exe /k "mvn install:install-file -Dfile=D:\Projects\Laptop\_Project\MyBank\mybankdatabase\json-simple-1.1.1.jar -DgroupId=com.googlecode.json-simple -DartifactId=json-simple -Dversion=1.1.1 -Dpackaging=jar -DgeneratePom=true"
289. echo "INSTALLING jeventClient jar files maven repository .. "
290. start "MVN INSTALLING jeventClient jar files maven repository .. " cmd.exe /k "mvn install:install-file -Dfile=D:\Projects\Laptop\_Project\MyBank\mybankeventclient\jevent.jar -DgroupId=myevent -DartifactId=jevent -Dversion=1.0 -Dpackaging=jar -DgeneratePom=true"
291. echo "TESTING jeventClient connection .."
292. echo
293. mvn exec:java
294. So we will just see what basically our project do … We have 4 main parts eg Customer, Accounts, Event Sync, Money Transfer…
295. In Customer format you Details about Customer here you can can perform CRUD method (CREATE READ UPDATE DELETE) the data…
296. Account also give similar to Customer… you can view account details…
297. Now the main work is done by EVENT SYNC… event sync does synchronise Customer and Account and database together.. suppose you edit or add account in Customer section then it should be reflect on Account webpage too…
298. This is basic function of your project….If you have any doubts please refer to sirs 1st video (only starting 15 min part then you will understand it … )
299. If you want to understand which code does what you can refer video 1,2
300. To make UML Diagram Refer Video 3…

----------------------------------------THANK YOU----------------------------------------