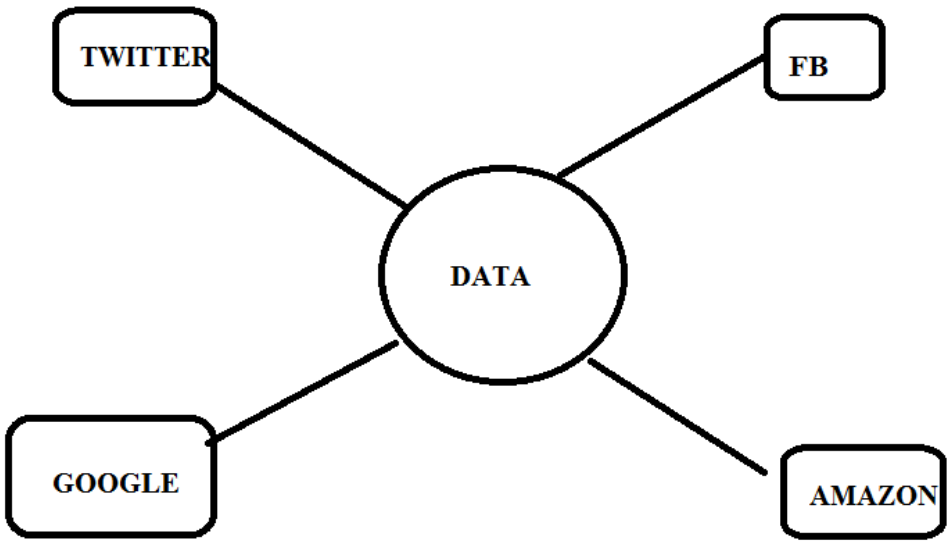
JDBC - Java DataBase Connectivity

Class Notes on 21-June-2021

https://docs.google.com/document/d/1VahBcOI2MqwxZrZl9k6uvK6KtUJk0CpjSaJJWXr3Jd0/edit?usp=sharing

JDBC

Java Data Base Connectivity



We need JDBC Driver to connect to DATABASE servers
We need specific JDBC Driver to connect to SPECIFIC DATABASE servers

You can't use the same JDBC Driver to connect to different DB servers

Examples:

Use separate JDBC driver to connect to ORACLE DB
Use separate JDBC driver to connect to MONGODB
Use Separate JDBC driver to connect to GraphDB

Who must provide this JDBC driver?

JDBC Driver is a Java Class, special Java class
It must be provided by the DB vendors

Using this, we can connect to specific DB servers

Where can I find this driver class?

In "lib" folder of ORACLE installation location, you can see a file with the name "ojdbc5.jar"

In which format, this driver class be available?

The JDBC driver class is available in the form of .jar file

CREATE TABLE BOOK(BOOKID INTEGER, BOOKNAME VARCHAR2(20), AUTHOR VARCHAR2(20), BOOKPRICE NUMBER(10,2))

Creating a Project for working with JDBC:

1. Create a Java Project and give the name "JDBCDemo1"

```
SQL> select * from books;

    BOOKID BOOKNAME            AUTHOR               BOOKPRICE
---------- -------------------- -------------------- ----------
         1 Java Programming     Guru                        750
         2 Cloud Computing      Stephen                    9950
         3 DevOps Basics        Peter                       350
         4 GraphDB Basics       Anil                        625
         5 Neural Networks      Choudry                    2625

SQL>
```

11.30
Be back at 11.45

Exercise:
Write separate JDBC program for achieving the below:
a)

Select all books written by the Author "Guru"
SELECT BOOKNAME FROM BOOKS WHERE AUTHOR='Guru';

b)

Display all books written by the Author who has written the book about "Java"
SELECT * FROM BOOKS WHERE AUTHOR IN(SELECT * FROM BOOKS WHERE BOOKNAME LIKE '%Java%)

10 minutes

We have learnt the following JDBC API:

1. **DriverManager.getConnection()**
2. **createStatement() to create Statement object**
3. **executeQuery() method to execute SQL queries (only for SELECT statements)**
4. **next() method to move the logical cursor to successive rows**
5. **How to use getString(), getInt(), getDouble() methods**

java.sql package contains only the interfaces
But, the ORACLE DB vendor (JDBC DRIVER -ojdbc5.jar) provides/contains all the implementations for this interface

To change the DB server, for instance, you have MySQL
Then, it will be sufficient to change the driver name and connection details, that's all

Exercise 2:

How to insert a row from Java program?
You can use Statement.

Exercise 3:

How to enter dynamic values into a TABLE ROW?
You should use PreparedStatement

Time: 1 pm
Be back at 2 pm

You can implement CRUD operations
C - CREATE
R - READ
U - UPDATE
D - DELETE

Using executeUpdate() method, you can implement CUD operations in CRUD
Using executeQuery() method, you can implement R operations in CRUD

Exercise 3

Get a book id from the user for deletion (1)

Display the details about the book

1 Java Guru 123.5 (SELECT - executeQuery)

Get a confirmation from the user whether to go for deletion or not

If the user confirms, delete (DELETE - executeUpdate)

Else don't delete

Just get only one DB connection

Using this you must do both the operations

15 mins

2.13

2.28 pm

Similarly, if you want to update, you should use UPDATE sql query:

Example:

I want to increase the book price by **xxx** rupees for all the books written by **yyy**

Code snippet:

PreparedStatement psmt=con.prepareStatement("update books set bookprice=bookprice+? Where author=?");

Accept the hike value and the author name from the user as input

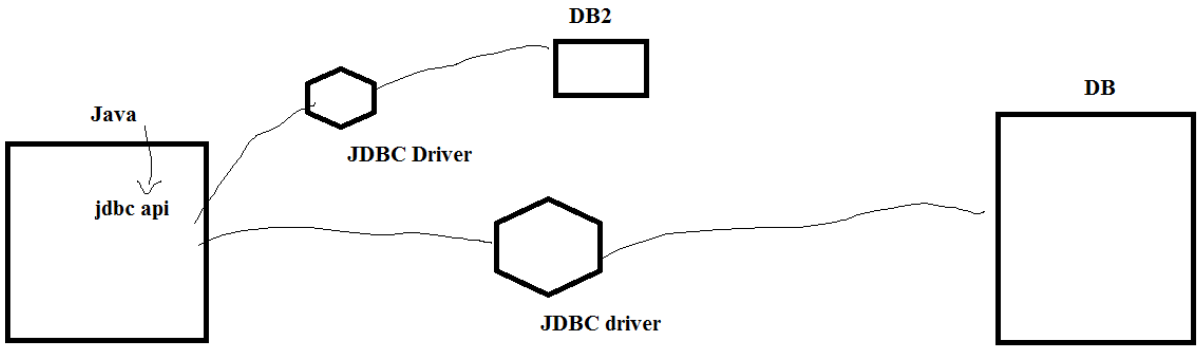
psmt.setDouble(1,hikePrice); // one variable
psstmt.setString(2,author); // input

int reccnt=pstmt.executeUpdate();
If (reccnt>0)

SOP("updated successfully");

Take 10 minutes more to complete the update operation

Time: 2.54
Finish by: 3.05



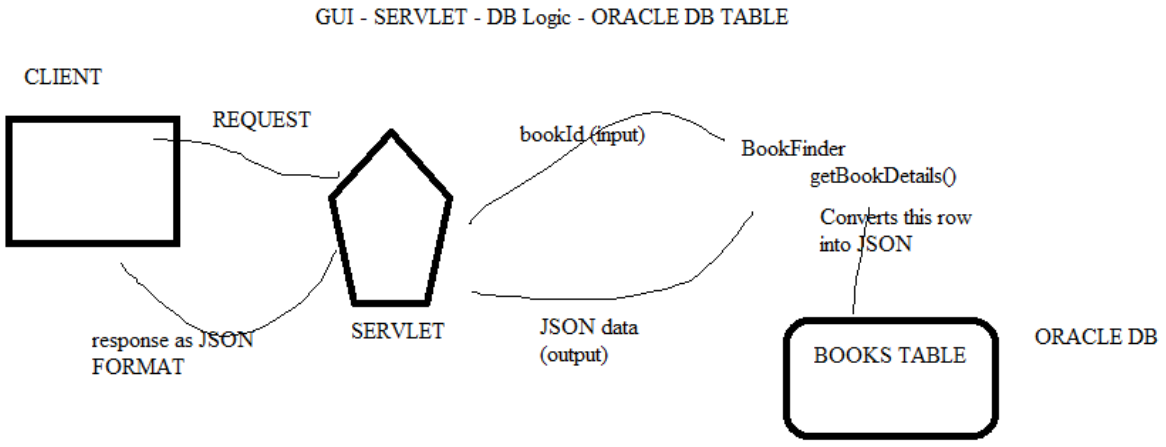
With DATA, the general operations are CRUD
Using this, you can manage any type of resources such as
BOOKS, EMPLOYEES, STUDENTS, SONGS, MOVIES….
This is sometimes called as RESOURCE MANAGEMENT

How can I use JDBC in my web app / servlet applications?

Exercise 4:

Get the book id from the user in HTML page
Send this book id to a servlet
Let the servlet connect to DB table and fetch the details and send it back to the user in JSON format

I am integrating WebApplication with Database server



```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h1> Welcome to My Book Finder Page</h1>
<h2> Enter the details here</h2>

<form action="BookFinderServlet">
Enter bookid: <input type="number" name="bid"><br>
<input type="submit" value="Search the book"/>
</form>
</body>
</html>
```

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
		// TODO Auto-generated method stub
		PrintWriter pw=response.getWriter();
		response.setContentType("text/html");
		int bookId=Integer.parseInt(request.getParameter("bid"));
		// Here, you can include JDBC code
		BookFinder bf=new BookFinder();
		Book bObj=null;
		try {
			bObj = bf.getBookDetails(bookId);
		} catch (Exception e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
		// Now, you can decide in which format you want to send?
		// XML or JSON format
		Gson gsonObj=new Gson();
		String jsonObj=gsonObj.toJson(bObj);
		pw.println(jsonObj);

	}
```

```
package com.pwc;

import java.sql.*;
public class BookFinder {
```

```
	Connection con=null;


	// utility method to get DB connection

	public Connection getConnection() throws Exception
	{
		Class.forName("oracle.jdbc.driver.OracleDriver");
		if (con==null)

con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","Password12
345");

		return con;
	}

	//Method to return book details for a given book id
	public Book getBookDetails(int bookId) throws Exception
	{
		getConnection();
		String sql="SELECT * FROM BOOKS WHERE BOOKID=?";
		PreparedStatement pstmt=con.prepareStatement(sql);
		pstmt.setInt(1, bookId);
		ResultSet rs=pstmt.executeQuery();
		rs.next();
		int id=rs.getInt(1);
		String bookName=rs.getString(2);
		String author=rs.getString(3);
		double price=rs.getDouble(4);
		// For this particular row, let me create an object of Book class
		Book b=new Book(id, bookName, author, price);
		return b;


	}


}


package com.pwc;

// POJO class
```

```
public class Book {
	int bookId;
	String bookName;
	String authorName;
	double bookPrice;

	public Book() {
		super();
		// TODO Auto-generated constructor stub
	}

	public Book(int bookId, String bookName, String authorName, double bookPrice) {
		super();
		this.bookId = bookId;
		this.bookName = bookName;
		this.authorName = authorName;
		this.bookPrice = bookPrice;
	}

	public int getBookId() {
		return bookId;
	}

	public void setBookId(int bookId) {
		this.bookId = bookId;
	}

	public String getBookName() {
		return bookName;
	}

	public void setBookName(String bookName) {
		this.bookName = bookName;
	}

	public String getAuthorName() {
		return authorName;
	}

	public void setAuthorName(String authorName) {
		this.authorName = authorName;
	}

	public double getBookPrice() {
```

```
		return bookPrice;
	}

	public void setBookPrice(double bookPrice) {
		this.bookPrice = bookPrice;
	}


}
```

-- You take 20 minutes to complete this exercise

3.40

Be back: 3.55
Complete by: 4.05

The above should be done in a dynamic web project
Web application -
HTML + SERVLET + Normal Java Class (Business Logic class) + POJO class + ORACLE DB

In this program, what we got from the servlet is just a RESPONSE

The response in JSON/XML format is of NO USE to the end client
Instead, we should return **VIEW** that end client can understand

This is called AS **VIEW**

Book Id	Title	Author	Price
9	Python	SOMEBODY	502.0

Anything you send / return from the servlet that the end user finds it very easy to understand is called as VIEW
This view could be in any format such as:

1. Html table
2. Chart (bar chart, pie chart…)
3. Animation
4. Visual Picture

In all the cases, the DATA REMAINS THE SAME

We should understand MVC design pattern

M - Model (anything related to the data can go into this layer)

Books table and JDBC code to fetch data from from this table

Should not take any additional responsibility of creating view

V - View (anything related to the presentation can go into this layer)

We haven't done yet (HTML table)

Responsible for generating the views

But, should not take any additional responsibility of having the model code

C - Controller (anything related to controlling the activities of the application can go into this layer)

The Servlet

Responsible for controlling the activities of the application

But, should not take any additional responsibility of creating view or code related to model layer

We can break an application into the above 3 layers

We can create view related code in Servlets (this is technically possible)
We should not create view related code in Servlets (this is design-wise wrong)
I am not going to write view related code in Servlets

JSP - Java Server Pages
JSP = HTML + Java code

Let's see how the servlet gives the actual value to JSP so that JSP can format the result
---

```
package com.pwc;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.google.gson.Gson;

/**
 * Servlet implementation class BookFinderServlet
```

```
 */
public class BookFinderServlet extends HttpServlet {
	private static final long serialVersionUID = 1L;

   /**
    * @see HttpServlet#HttpServlet()
    */
   public BookFinderServlet() {
      super();
      // TODO Auto-generated constructor stub
   }

	/**
	 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
	 */
	protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
		// TODO Auto-generated method stub
		PrintWriter pw=response.getWriter();
		response.setContentType("text/html");
		int bookId=Integer.parseInt(request.getParameter("bid"));
		// Here, you can include JDBC code
		BookFinder bf=new BookFinder();
		Book bObj=null;
		try {
			bObj = bf.getBookDetails(bookId);
		} catch (Exception e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
		// Now, you can decide in which format you want to send?
		// XML or JSON format
//		Gson gsonObj=new Gson();
//		String jsonObj=gsonObj.toJson(bObj);
//		pw.println(jsonObj);
		// Here, I will be sending the view

		if (bObj==null)
		{
			bObj=new Book(); // create an empty book object
			bObj.setBookId(bookId);
			bObj.setBookName("No book found for this id "+bookId);
			bObj.setAuthorName("");
			bObj.setBookPrice(0.0);
```

```
			}
			request.setAttribute("bookObject", bObj);
			request.getRequestDispatcher("BooksView.jsp").forward(request, response);

		}

		/**
		 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
		 */
		protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
			// TODO Auto-generated method stub
			doGet(request, response);
		}

}
```

BooksView.jsp

```
<%@page import="com.pwc.*" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>

<%
		Book bOjb=(Book)request.getAttribute("bookObject");
 %>
<h1>Here are your details:</h1>
<table border="2">
		<tr>
			<td>Book Id</td>
			<td>Title</td>
			<td>Author</td>
			<td>Price</td>
		</tr>

		<tr>

			<td> <%=bOjb.getBookId()%></td>
```

```
                    <td> <%=bOjb.getBookName() %></td>
                    <td> <%=bOjb.getAuthorName() %></td>
                    <td> <%=bOjb.getBookPrice() %></td>

            </tr>
</table>

<a href="BookFinder.htm">Search another book</a>
</body>
</html>
```