```
Create Employee Functionality
  a. ENDPOINTS : POST -> http://localhost:9999/api/v1/employees


React
+++++
1. Create a button in ListEmployeeComponent to perform Insert operation

ListEmployeeComponent.jsx
+++++++++++++++++++++++++
import React, { useState,useEffect } from 'react';
import {getEmployees} from "../services/EmployeeService"
import { useNavigate } from 'react-router-dom';



function ListEmployeeComponent() {

  //employees <----setEmployees()
  const [employees, setEmployees] = useState([]);
  const navigate = useNavigate();


  useEffect(() => {
    //Calling API From BackEnd to get the data of Employees
    getEmployees().then((emp) => {
     //Inject data to employees varaible
      setEmployees(emp.data);
    });
  }, []);

  //navigation to the respective component
  const addEmployee =()=> {
    navigate("/add-employee")
  }

  return (
    <div className="m-4">
      <h2 className="text-center">Employees List</h2>

      <!-- CREATING AN ADD BUTTON TO ADD EMPLOYEE RECORD -->
      <div className = "row">
        <button className='btn btn-primary' onClick={addEmployee}>ADD
EMPLOYEE</button>
      </div><br/>

      <div className="row">
        <table className="table table-striped table-bordered">
          <thead>
            <tr>
              <th> Employee First Name</th>
              <th> Employee Last Name</th>
              <th> Employee Email Id</th>
              <th> Actions</th>
            </tr>
          </thead>
          <tbody>
            {employees.map((employee) => (
              <tr key={employee.id}>
```

```
                <td> {employee.firstName}</td>
                <td> {employee.lastName}</td>
                <td> {employee.emailId}</td>
              </tr>
            ))}
          </tbody>
        </table>
      </div>
    </div>
  );
}
export default ListEmployeeComponent;
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++

Configure Routing in App.js page
++++++++++++++++++++++++++++++++
import ListEmployeeComponent from './component/ListEmployeeComponent';
import HeaderComponent from './component/HeaderComponent';
import FooterComponent from './component/FooterComponent';
import { BrowserRouter as Router } from 'react-router-dom';
import { Routes, Route } from "react-router";

import 'bootstrap/dist/css/bootstrap.min.css'
import './App.css';
import CreateEmployeeComponent from './component/CreateEmployeeComponent';

function App() {
  return (
    <div>
      <Router>
        <HeaderComponent />
            <div class="container">
            <Routes>
                  <Route path='/' element={<ListEmployeeComponent />} />
                  <Route path='/employees' element={<ListEmployeeComponent />} />

                  <!-- Configured Routing to route for /add-employee which display
insert record page --->
                  <Route path='/add-employee' element={<CreateEmployeeComponent />}
/>
            </Routes>
          </div>
        <FooterComponent />
      </Router>
    </div>
  );
}

export default App;

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Create a Component called CreateEmployeeComponent in src folder

+++++++++++++++++++++++++++
CreateEmployeeComponent.jsx
+++++++++++++++++++++++++++
import React, { useState } from 'react';
import { useNavigate } from 'react-router-dom';

function CreateEmployeeComponent() {

 //React hook which will be used to render the page
  const [employee, setEmployee] = useState({
    firstName: '',
    lastName: '',
    emailId: ''
  });

  const navigate = useNavigate()

  const changeFirstNameHandler = (event) => {
    setEmployee({ ...employee, firstName: event.target.value });
  }

  const changeLastNameHandler = (event) => {
    setEmployee({ ...employee, lastName: event.target.value });
  }

  const changeEmailHandler = (event) => {
    setEmployee({ ...employee, emailId: event.target.value });
  }

  //Performing necessary actions
  const saveOrUpdateEmployee = (e) => {
    //Display the output values on the console to check whether it is coming or not
    e.preventDefault();
    console.log('employee => ' + JSON.stringify(employee));
  }

  //Performing necessary actions
  const cancel = () => {
    navigate("/")
  }

  return (
    <div>
      <br></br>
      <div className="container">
        <div className="row">
          <div className="card col-md-6 offset-md-3 offset-md-3">
            <h3 className="text-center">Add Employee</h3>
            <div className="card-body">
              <form>

                <div className="form-group">
                  <label> First Name: </label>
                  <input placeholder="First Name" name="firstName" className="form-
control"
                    value={employee.firstName} onChange={changeFirstNameHandler} />
                </div>

                <div className="form-group">
                  <label> Last Name: </label>
                  <input placeholder="Last Name" name="lastName" className="form-
control"
```

```
                    value={employee.lastName} onChange={changeLastNameHandler} />
                </div>

                <div className="form-group">
                  <label> Email Id: </label>
                  <input placeholder="Email Address" name="emailId"
className="form-control"
                    value={employee.emailId} onChange={changeEmailHandler} />
                </div>

                <button className="btn btn-success"
onClick={saveOrUpdateEmployee}>Save</button>
                <button className="btn btn-danger" onClick={cancel}
style={{ marginLeft: "10px" }}>Cancel</button>

              </form>
            </div>
          </div>
        </div>
      </div>
    </div>
  );
}

export default CreateEmployeeComponent;
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++
 Upon clicking on ADD EMPLOYEE -> CreateEmploye Component page will be available
    upon entering the details and clicking on save ---> it should make a call to
      ENDPOINTS : POST -> http://localhost:9999/api/v1/employees

Create a new Service in ServiceComponent.jsx
++++++++++++++++++++++++++++++++++++++++++++
import axios from 'axios';

const EMPLOYEE_API_BASE_URL = "http://localhost:9999/api/v1/employees";

export const getEmployees = () => {
  return axios.get(EMPLOYEE_API_BASE_URL);
};

export const createEmployee = (employee) => {
  return axios.post(EMPLOYEE_API_BASE_URL, employee);
};
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

CreateEmployeeComponent.jsx
+++++++++++++++++++++++++++
import React, { useState } from 'react';
import { useNavigate } from 'react-router-dom';
import { createEmployee } from '../services/EmployeeService';



function CreateEmployeeComponent() {

 //React hook which will be used to render the page
  const [employee, setEmployee] = useState({
    firstName: '',
    lastName: '',
    emailId: ''
  });

  const navigate = useNavigate()

  const changeFirstNameHandler = (event) => {
    setEmployee({ ...employee, firstName: event.target.value });
  }

  const changeLastNameHandler = (event) => {
    setEmployee({ ...employee, lastName: event.target.value });
  }

  const changeEmailHandler = (event) => {
    setEmployee({ ...employee, emailId: event.target.value });
  }

  const saveOrUpdateEmployee = (e) => {
    //Display the output values on the console to check whether it is coming or not
     e.preventDefault();
    console.log('employee => ' + JSON.stringify(employee));

    //Actual code which will call the services from backend and re-direct to main
page
    createEmployee(employee).then((res) => {
        navigate('/employees'); // Use navigate
      });
  }

  const cancel = () => {
    navigate("/")
  }

  return (
    <div>
      <br></br>
      <div className="container">
        <div className="row">
          <div className="card col-md-6 offset-md-3 offset-md-3">
            <h3 className="text-center">Add Employee</h3>
            <div className="card-body">
              <form>
                <div className="form-group">
                  <label> First Name: </label>
                  <input placeholder="First Name" name="firstName" className="form-
control"
                    value={employee.firstName} onChange={changeFirstNameHandler} />
                </div>
                <div className="form-group">
                  <label> Last Name: </label>
                  <input placeholder="Last Name" name="lastName" className="form-
control"
                    value={employee.lastName} onChange={changeLastNameHandler} />
                </div>
                <div className="form-group">
                  <label> Email Id: </label>
                  <input placeholder="Email Address" name="emailId"
className="form-control"
```

```
                    value={employee.emailId} onChange={changeEmailHandler} />
                </div>

                <button className="btn btn-success"
onClick={saveOrUpdateEmployee}>Save</button>
                <button className="btn btn-danger" onClick={cancel}
style={{ marginLeft: "10px" }}>Cancel</button>
              </form>
            </div>
          </div>
        </div>
      </div>
    </div>
  );
}

export default CreateEmployeeComponent;

+++++++++++++++++++++++++++++++++++++++++++OUTPUT DISPLAYED(ListEmployeeComponent)+
++++++++++++++++++++++++++++++++++++++++++++++++++++++++


Working with Update Module
++++++++++++++++++++++++++
 1. Create a ENDPOINT to get the record based on id value
      GET :: http://localhost:9999/api/v1/employees/{id}

 2. Create a React code to send the request to the page

a. Update ListEmployeeComponent.jsx with UPDATE Button
import React, { useState,useEffect } from 'react';
import {getEmployees} from "../services/EmployeeService"
import { useNavigate } from 'react-router-dom';

function ListEmployeeComponent() {

  //employees <----setEmployees()
  const [employees, setEmployees] = useState([]);
  const navigate = useNavigate();


  useEffect(() => {
    //Calling API From BackEnd to get the data of Employees
    getEmployees().then((emp) => {
     //Inject data to employees varaible
      setEmployees(emp.data);
    });
  }, []);


  const addEmployee =()=> {
    navigate("/add-employee")
  }

  //COLLECTED THE DYNAMIC VALUE BASED ON THE USER INTERACTION
  const editEmployee = (id) => {
    console.log("id:: " + id);

    //USE ROUTER TO ROUTE TO /update-employee/${id} ----> UpdateEmployeeComponent
    navigate(`/update-employee/${id}`);
  };

  return (
    <div className="m-4">
      <h2 className="text-center">Employees List</h2>
      <div className = "row">
        <button className='btn btn-primary' onClick={addEmployee}>ADD
EMPLOYEE</button>
      </div><br/>

      <div className="row">
        <table className="table table-striped table-bordered">
          <thead>
            <tr>
              <th> Employee First Name</th>
              <th> Employee Last Name</th>
              <th> Employee Email Id</th>
              <th> Actions</th>
            </tr>
          </thead>
          <tbody>
            {employees.map((employee) => (
              <tr key={employee.id}>
                <td> {employee.firstName}</td>
                <td> {employee.lastName}</td>
                <td> {employee.emailId}</td>

                 <!-- CREATING A BUTTON FOR UPDATE --!>
                <button onClick={() => editEmployee(employee.id)} className="btn
btn-info">Update</button>

              </tr>
            ))}
          </tbody>
        </table>
      </div>
    </div>
  );
}
export default ListEmployeeComponent;
```

++
Create an UpdateEmployeeComponent.jsx in src folder

```
import React, { useState, useEffect } from 'react';
import { useParams, useNavigate } from 'react-router-dom';
import { getEmployeeById, updateEmployee } from '../services/EmployeeService';

function UpdateEmployeeComponent() {

    const navigate = useNavigate()
    const { id } = useParams();

    const [employee, setEmployee] = useState({
        id: id,
        firstName: '',
        lastName: '',
        emailId: ''
    });

    useEffect(() => {
        const fetchEmployee = async () => {
            try {
                //Calling the Backend api to load the data to the page based on id
value
                const response = await getEmployeeById(id);
                const employeeData = response.data;
                setEmployee({
                    id: id,
                    firstName: employeeData.firstName,
                    lastName: employeeData.lastName,
                    emailId: employeeData.emailId,
                });
            } catch (error) {
                console.error(error);
            }
        };

        fetchEmployee();
    }, [id]);

    const updateEmployeeData = async (e) => {
        e.preventDefault();
        console.log(employee)
    };


    const changeFirstNameHandler = (event) => {
        setEmployee({ ...employee, firstName: event.target.value });
    }

    const changeLastNameHandler = (event) => {
        setEmployee({ ...employee, lastName: event.target.value });
    }

    const changeEmailHandler = (event) => {
        setEmployee({ ...employee, emailId: event.target.value });
    }


    const changeHandler = (event) => {
        setEmployee({
            ...employee,
            [event.target.name]: event.target.value,
        });
    };


    const cancel = () => {
        navigate("/")
    }

    return (
        <div>
            <br></br>
            <div className="container">
```

```
                <div className="row">
                    <div className="card col-md-6 offset-md-3 offset-md-3">
                        <h3 className="text-center">Update Employee</h3>
                        <div className="card-body">
                            <form>

                                <div className="form-group">
                                    <label> First Name: </label>
                                    <input placeholder="First Name"
name="firstName" className="form-control"
                                        value={employee.firstName}
onChange={changeFirstNameHandler} />
                                </div>

                                <div className="form-group">
                                    <label> Last Name: </label>
                                    <input placeholder="Last Name" name="lastName"
className="form-control"
                                        value={employee.lastName}
onChange={changeLastNameHandler} />
                                </div>
                                <div className="form-group">
                                    <label> Email Id: </label>
                                    <input placeholder="Email Address"
name="emailId" className="form-control"
                                        value={employee.emailId}
onChange={changeEmailHandler} />
                                </div>

                                <button className="btn btn-success"
onClick={updateEmployeeData}>update</button>
                                <button className="btn btn-danger" onClick={cancel}
style={{ marginLeft: "10px" }}>Cancel</button>
                            </form>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    );
}
export default UpdateEmployeeComponent;
```

++
Create a Routing to /update-employee/{id} in App.js

App.js
+++++

```
import ListEmployeeComponent from './component/ListEmployeeComponent';
import HeaderComponent from './component/HeaderComponent';
import FooterComponent from './component/FooterComponent';
import { BrowserRouter as Router } from 'react-router-dom';
import { Routes, Route } from "react-router";

import 'bootstrap/dist/css/bootstrap.min.css'
import './App.css';
import CreateEmployeeComponent from './component/CreateEmployeeComponent';
import UpdateEmployeeComponent from './component/UpdateEmployeeComponent';
```

```
function App() {
  return (
    <div>
      <Router>
        <HeaderComponent />
            <div class="container">
            <Routes>
                  <Route path='/' element={<ListEmployeeComponent />} />
                  <Route path='/employees' element={<ListEmployeeComponent />} />
                  <Route path='/add-employee' element={<CreateEmployeeComponent />}
/>

                  <!-- ROUTING FOR UPDATING THE RECORD --!>
                  <Route path='/update-employee/:id' element={<
UpdateEmployeeComponent/>} />
            </Routes>
          </div>
        <FooterComponent />
      </Router>
    </div>
  );
}

export default App;

+++++++++++++++++++++++++++++++++++++Page rendered based on the id value selected by
the user ++++++++++++++++++++++++++++++++++++++++++++++++
 => User can make changes to the data present in the text box.
 => Upon clicking the save button respective changes should be made in the database
by making a call to backend.

Update END POINT :: PUT -> http://localhost:9999/api/v1/employees/{id}

App.js
+++++++
import axios from 'axios';

const EMPLOYEE_API_BASE_URL = "http://localhost:9999/api/v1/employees";

export const getEmployees = () => {
  return axios.get(EMPLOYEE_API_BASE_URL);
};

export const createEmployee = (employee) => {
  return axios.post(EMPLOYEE_API_BASE_URL, employee);
};

export function getEmployeeById(employeeId) {
  return axios.get(`${EMPLOYEE_API_BASE_URL}/${employeeId}`);
}

export function updateEmployee(employee, employeeId) {
  return axios.put(`${EMPLOYEE_API_BASE_URL}/${employeeId}`, employee);
}
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++
UpdateEmployeeComponent.jsx

import React, { useState, useEffect } from 'react';
import { useParams, useNavigate } from 'react-router-dom';
import { getEmployeeById,updateEmployee} from '../services/EmployeeService';

function UpdateEmployeeComponent() {

    const navigate = useNavigate()
    const { id } = useParams();

    const [employee, setEmployee] = useState({
        id: id,
        firstName: '',
        lastName: '',
        emailId: ''
    });

    useEffect(() => {
        const fetchEmployee = async () => {
            try {
                //Calling the Backend api to load the data to the page based on id
value
                const response = await getEmployeeById(id);
                const employeeData = response.data;
                setEmployee({
                    id: id,
                    firstName: employeeData.firstName,
                    lastName: employeeData.lastName,
                    emailId: employeeData.emailId,
                });
            } catch (error) {
                console.error(error);
            }
        };

        fetchEmployee();
    }, [id]);


    const updateEmployeeData = async (e) => {
        e.preventDefault();
        console.log(employee)

        //Calling API to send the data to BackEnd to perform update operation
        await updateEmployee(employee, id);
        navigate('/employees');
    };


    const changeFirstNameHandler = (event) => {
        setEmployee({ ...employee, firstName: event.target.value });
    }

    const changeLastNameHandler = (event) => {
        setEmployee({ ...employee, lastName: event.target.value });
    }

    const changeEmailHandler = (event) => {
        setEmployee({ ...employee, emailId: event.target.value });
    }
    const changeHandler = (event) => {
        setEmployee({
            ...employee,
            [event.target.name]: event.target.value,
        });
    };


    const cancel = () => {
        navigate("/")
    }

    return (
        <div>
            <br></br>
            <div className="container">
                <div className="row">
                    <div className="card col-md-6 offset-md-3 offset-md-3">
                        <h3 className="text-center">Update Employee</h3>
                        <div className="card-body">
                            <form>

                                <div className="form-group">
                                    <label> First Name: </label>
                                    <input placeholder="First Name"
name="firstName" className="form-control"
                                        value={employee.firstName}
onChange={changeFirstNameHandler} />
                                </div>

                                <div className="form-group">
                                    <label> Last Name: </label>
                                    <input placeholder="Last Name" name="lastName"
className="form-control"
                                        value={employee.lastName}
onChange={changeLastNameHandler} />
                                </div>
                                <div className="form-group">
                                    <label> Email Id: </label>
                                    <input placeholder="Email Address"
name="emailId" className="form-control"
                                        value={employee.emailId}
onChange={changeEmailHandler} />
                                </div>

                                <button className="btn btn-success"
onClick={updateEmployeeData}>update</button>
                                <button className="btn btn-danger" onClick={cancel}
style={{ marginLeft: "10px" }}>Cancel</button>
                            </form>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    );
}
export default UpdateEmployeeComponent;
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++

Delete record based on the id value
+++++++++++++++++++++++++++++++++++
 1. Create an ENDPOINT
      DELETE: http://localhost:9999/api/v1/employees/{id}

 2. React
      a. Create a button called delete in ListEmployeeComponent.jsx

import React, { useState,useEffect } from 'react';
import {getEmployees} from "../services/EmployeeService"
import { useNavigate } from 'react-router-dom';



function ListEmployeeComponent() {

  //employees <----setEmployees()
  const [employees, setEmployees] = useState([]);
  const navigate = useNavigate();


  useEffect(() => {
    //Calling API From BackEnd to get the data of Employees
    getEmployees().then((emp) => {
     //Inject data to employees varaible
      setEmployees(emp.data);
    });
  }, []);


  const addEmployee =()=> {
    navigate("/add-employee")
  }

  const editEmployee = (id) => {
    console.log("id:: " + id);
    navigate(`/update-employee/${id}`);
  };

  //logic for deletion to happen upon the click
  const deleteEmployee = (id) => {
    console.log("id:: " + id);
  };

  return (
    <div className="m-4">
      <h2 className="text-center">Employees List</h2>
      <div className = "row">
        <button className='btn btn-primary' onClick={addEmployee}>ADD
EMPLOYEE</button>
      </div><br/>

      <div className="row">
        <table className="table table-striped table-bordered">
          <thead>
```

```
            <tr>
              <th> Employee First Name</th>
              <th> Employee Last Name</th>
              <th> Employee Email Id</th>
              <th> Actions</th>
            </tr>
          </thead>
          <tbody>
            {employees.map((employee) => (
              <tr key={employee.id}>
                <td> {employee.firstName}</td>
                <td> {employee.lastName}</td>
                <td> {employee.emailId}</td>
                <button onClick={() => editEmployee(employee.id)} className="btn
btn-info">Update</button>

                  <!-- DELETE BUTTON ---- !>
                <button onClick={() => deleteEmployee(employee.id)} className="btn
btn-danger ml-2">Delete</button>

              </tr>
            ))}
          </tbody>
        </table>
      </div>
    </div>
  );
}
export default ListEmployeeComponent;
```

++
++++++

 b. Add the services for delete in EmployeeService.jsx

```
import axios from 'axios';

const EMPLOYEE_API_BASE_URL = "http://localhost:9999/api/v1/employees";

export const getEmployees = () => {
  return axios.get(EMPLOYEE_API_BASE_URL);
};

export const createEmployee = (employee) => {
  return axios.post(EMPLOYEE_API_BASE_URL, employee);
};

export function getEmployeeById(employeeId) {
  return axios.get(`${EMPLOYEE_API_BASE_URL}/${employeeId}`);
}

export function updateEmployee(employee, employeeId) {
  return axios.put(`${EMPLOYEE_API_BASE_URL}/${employeeId}`, employee);
}

export function deleteEmployeeById(employeeId) {
  return axios.delete(`${EMPLOYEE_API_BASE_URL}/${employeeId}`);
}
```

 c. Make the changes in ListEmployeeComponent.jsx to delete the record based on id

```
value

import React, { useState,useEffect } from 'react';
import {getEmployees,deleteEmployeeById} from "../services/EmployeeService"
import { useNavigate } from 'react-router-dom';



function ListEmployeeComponent() {

  //employees <----setEmployees()
  const [employees, setEmployees] = useState([]);
  const navigate = useNavigate();


  useEffect(() => {
    //Calling API From BackEnd to get the data of Employees
    getEmployees().then((emp) => {
     //Inject data to employees varaible
      setEmployees(emp.data);
    });
  }, []);


  const addEmployee =()=> {
    navigate("/add-employee")
  }

  const editEmployee = (id) => {
    console.log("id:: " + id);
    navigate(`/update-employee/${id}`);
  };

  const deleteEmployee = (id) => {
    console.log("id:: " + id);

    //Make a call to backend to delete the record
    deleteEmployeeById(id).then((res) => {
       //display the same page with the record not matching with the supplied id
value
      setEmployees(employees.filter((employee) => employee.id !== id));
  });
};
return (
    <div className="m-4">
      <h2 className="text-center">Employees List</h2>
      <div className = "row">
        <button className='btn btn-primary' onClick={addEmployee}>ADD
EMPLOYEE</button>
      </div><br/>

      <div className="row">
        <table className="table table-striped table-bordered">
          <thead>
            <tr>
              <th> Employee First Name</th>
              <th> Employee Last Name</th>
              <th> Employee Email Id</th>
              <th> Actions</th>
            </tr>
          </thead>
          <tbody>
            {employees.map((employee) => (
              <tr key={employee.id}>
                <td> {employee.firstName}</td>
                <td> {employee.lastName}</td>
                <td> {employee.emailId}</td>


                <button onClick={() => editEmployee(employee.id)} className="btn
btn-info">Update</button>
                <button onClick={() => deleteEmployee(employee.id)} className="btn
btn-danger ml-2">Delete</button>

              </tr>
            ))}
          </tbody>
        </table>
      </div>
    </div>
  );
}
export default ListEmployeeComponent;
```

++++++++++++++++++++++++++++++++++Fully Functional Code is Ready++++++++++++++++++
++

Server Managed Connection Pool
++++++++++++++++++++++++++++++
Connection Pooling Management(CPM)
  => It is a new way of organising connection objects in JavaEE environment.
  => Connection Pooling manager is a software component and it is a part of server and it manages connection objects.
  => Here the connection objects are reusable.

To configure connection pool, use the following
 a. Open the browser
 b. Open google
 c. Type the following in search bar(tomcat resource tag)

 d. Click on the following link
      https://tomcat.apache.org/tomcat-9.0-doc/jndi-resources-howto.html
 e. Click on JDBC datasources hyperlink

 f. Copy Resource tag

```
      <Resource name="jdbc/EmployeeDB"
            auth="Container"
            type="javax.sql.DataSource"
            username="dbusername"
            password="dbpassword"
            driverClassName="org.hsql.jdbcDriver"
            url="jdbc:HypersonicSQL:database"
            maxTotal="8"
            maxIdle="4"/>
```

 g. Copied text paste in context.xml file between <Context> tags.
      In eclipse environment goto servers folder
            open context.xml and paste the above code inside <context> tag.

```
 h. Change the following fields as per the db usage in project
      <Resource name="jdbc/EmployeeDB"
            auth="Container"
            type="javax.sql.DataSource"
            username="root"
            password="root123"
            driverClassName="com.mysql.cj.jdbc.Driver"
            url="jdbc:mysql://localhost:3306/octbatch"
            maxTotal="8"
            maxIdle="4"/>

 i. Access this connection pool in Servlet code.

// Obtain our environment naming context
Context initCtx = new InitialContext();
Context envCtx = (Context) initCtx.lookup("java:comp/env");

// Look up our data source
DataSource ds = (DataSource)envCtx.lookup("jdbc/EmployeeDB");

// Allocate and use a connection from the pool
Connection conn = ds.getConnection();

//use connection object to perform neccessary persistence operation
      conn.......

//use this connection to access the database ...
conn.close();

++++++++++++++++++++++++++++++Demonstrating the connection pool of server using
JNDI ********************************
context.xml
+++++++++++
<Context>

      <Resource name="jndi"
            auth="Container"
            type="javax.sql.DataSource"
            username="root"
            password="root123"
            driverClassName="com.mysql.cj.jdbc.Driver"
            url="jdbc:mysql://localhost:3306/octbatch"
            maxTotal="8"
            maxIdle="4"/>


    <!-- Default set of monitored resources. If one of these changes, the    -->
    <!-- web application will be reloaded.                                   -->
    <WatchedResource>WEB-INF/web.xml</WatchedResource>
    <WatchedResource>WEB-INF/tomcat-web.xml</WatchedResource>
    <WatchedResource>${catalina.base}/conf/web.xml</WatchedResource>

    <!-- Uncomment this to disable session persistence across Tomcat restarts -->
    <!--
    <Manager pathname="" />
    -->
</Context>
```

```
TestServlet.java
++++++++++++++++
package in.ineuron.nitin.controller;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.SQLException;

import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.sql.DataSource;


public class TestServlet extends HttpServlet {
      private static final long serialVersionUID = 1L;

      @Override
      protected void doGet(HttpServletRequest request, HttpServletResponse 
response) throws ServletException, IOException {

            try {
                  Context initCtx = new InitialContext();
                  Context envCtx = (Context) initCtx.lookup("java:comp/env");

                  //Looking for JNDI registry with the name called "jndi"
                  DataSource ds = (DataSource)envCtx.lookup("jndi");

                  Connection conn = ds.getConnection();
                  PrintWriter out = response.getWriter();
                  out.println("<h1>Connection poool of server is :: 
"+conn+"</h1>");
                  out.close();
            } catch (NamingException | SQLException e) {
                  e.printStackTrace();
            }
      }

}

request  :http://localhost:8888/ConnectionPool-01/test
response :Connection poool of server is :: 1752069992, 
URL=jdbc:mysql://localhost:3306/octbatch, MySQL Connector/J
```

JNDI
+++
 => It stands for Java Naming Directory Interface
 => It is an API/technology available in jdksoftware that is given to interact with JNDI registery software.
 => Inside this JNDIRegistry we keep the datasource object reference.
 => JNDI Registry is given from server eniroment to keep the reference of datasource object which is available for every component in webapplication.
            refer: diagram

++++++++++++++++++++++
JDBC Driver information
++++++++++++++++++++++
Types of Drivers
While communicating with Database, we have to convert Java Calls into Database specific Calls and Database specific Calls into Java Calls.
For this Driver Software is required. In the Ma

rket Thousands of Driver Softwares are available.
But based on Functionality all Driver Software Drivers are divided into 4 Types.
1) Type-1 Driver (JDBC-ODBC Bridge Driver OR Bridge Driver)
2) Type-2 Driver (Native API-Partly Java Driver OR Native Driver)
3) Type-3 Driver (All Java Net Protocol Driver OR Network Protocol Driver OR Middleware Driver)
4) Type-4 Driver (All Java Native Protocol Driver OR Pure Java Driver OR Thin Driver)

This Driver provided by Sun Micro Systems as the Part of JDK. But this Support is available until 1.7 Version only.
Internally this Driver will take Support of ODBC Driver to communicate with Database.
Type-1 Driver converts JDBC Calls (Java Calls) into ODBC Calls and ODBC Driver converts ODBC Calls into Database specific Calls.
Hence Type-1 Driver acts as Bridge between JDBC and ODBC.

Advantages :
1. It is very easy to use and maintain.
2. We are not required to install any separate Software because it is available as the Part of JDK.
3. Type-1 Driver won't communicates directly with the Database. Hence it is Database Independent Driver. Because of this migrating from
   one Database to another Database will become Easy.

Limitations:
1. It is the slowest Driver among all JDBC Drivers (Snail Driver), because first it will convert JDBC Calls into ODBC Calls and ODBC Driver
   converts ODBC Calls into Database specific Calls.
2. This Driver internally depends on ODBC Driver, which will work only on Windows Machines. Hence Type-1 Driver is Platform Dependent Driver.
3. No Support from JDK 1.8 Version onwards.

Note:
Because of above Limitations it is never recommended to use Type-1 Driver.

+++++++++++++
Type-2 Driver
+++++++++++++
Type-2 Driver is exactly same as Type-1 Driver except that ODBC Driver is replaced with Vendor specific Native Libraries.
Type-2 Driver internally uses Vendor specific Native Libraries to Communicate with Database.
Native Libraries means the Set of Functions written in Non-Java (Mostly C OR C++).
We have to install Vendor provided Native Libraries on the Client Machine.
Type-2 Driver converts JDBC Calls into Vendor specific Native Library Calls, which can be understandable directly by Database Engine.

Advantages:
1. When compared with Type-1 Driver Performance is High, because it required only one Level Conversion from JDBC to Native Library Calls.
2. No need of arranging ODBC Drivers.
3. When compared with Type-1 Driver, Portability is more because Type-1 Driver is applicable only for Windows Machines.

Limitations:
1. Internally this Driver using Database specific Native Libraries and hence it is Database Dependent Driver.
   Because of this migrating from one Database to another Database will become Difficult.
2. This Driver is Platform Dependent Driver.
3. On the Client Machine compulsory we should install Database specific Native Libraries.
4. There is no Guarantee for every Database Vendor will provide This Driver.
(Oracle People provided Type-2 Driver but MySql People won't provide this Driver)
Eg: OCI (Oracle Call Interface) Driver is Type-2 Driver provided by Oracle.
OCI Driver internally uses OCI Libraries to communicate with Database.
OCI Libraries contain "C Language Functions"
OCI Driver and corresponding OCI Libraries are available in the following Jar File.
Hence we have to place this Jar File in the Class Path.
ojdbc14.jar ➔ Oracle 10g (Internally Uses Java 1.4V)
ojdbc6.jar ➔ Oracle 11g (Internally Uses Java 1.6V)
ojdbc7.jar ➔ Oracle 12c (Internally Uses Java 1.7V)

Note: The only Driver which is both Platform Dependent and Database Dependent is Type-2 Driver. Hence it is not recommended to use Type-2 Driver.

++++++++++++
Type-3 Driver
+++++++++++++

Type-3 Driver converts JDBC Calls into Middleware Server specific Calls. Middleware Server can convert Middleware Server specific Calls
into Database specific Calls.
Internally Middleware Server may use Type-1, 2 OR 4 Drivers to communicates with Database.

Advantages:
1. This Driver won't communicate with Database directly and hence it is Database Independent Driver.
2. This Driver is Platform Independent Driver.
3. No need of ODBC Driver OR Vendor specific Native Libraries.

Limitations:
1. Because of having Middleware Server in the Middle, there may be a chance of Performance Problems.
2. We need to purchase Middleware Server and hence the cost of this Driver is more when compared with remaining Drivers.
Eg: IDS Driver (Internet Database Access Server)
Note: The only Driver which is both Platform Independent and Database Independent is Type-3 Driver. Hence it is recommended to use.

+++++++++++++
Type-4 Driver

+++++++++++++
=> This Driver is developed to talk with the Database directly without taking Support of ODBC Driver OR Vendor Specific Native Libraries
   OR Middleware Server.
=> This Driver uses Database specific Native Protocols to communicate with the Database.
=> This Driver converts JDBC Calls directly into Database specific Calls.
=> This Driver developed only in Java and hence it is also known as Pure Java Driver. Because of this, Type-4 Driver is Platform Independent Driver.
=> This Driver won't require any Native Libraries at Client side and hence it is light weighted. Because of this it is treated as Thin Driver.

Advantages:
1. It won't require any Native Libraries, ODBC Driver OR Middleware Server
2. It is Platform Independent Driver
3. It uses Database Vendor specific Native Protocol and hence Security is more.

Limitation:
The only Limitation of this Driver is, it is Database Dependent Driver because it is communicating with the Database directly.
Eg: Thin Driver for Oracle, Connector/J Driver for MySQL

Note: It is highly recommended to use Type-4 Driver.
Java Application ➔ Type-1 Driver ➔ ODBC Driver ➔ DB
Java Application ➔ Type-2 Driver ➔ Vendor Specific Native Libraries ➔ DB
Java Application ➔ Type-3 Driver ➔ Middleware Server ➔ DB
Java Application ➔ Type-4 Driver ➔ DB

Which Driver should be used?
1. If we are using only one Type of Database in our Application then it is recommended to use Type-4 Driver.
Eg: Stand Alone Applications, Small Scale Web Applications

2. If we are using multiple Databases in our Application then Type-3 Driver is recommended to use.
Eg: Large Scale Web Applications and Enterprise Applications

3. If Type-3 and Type-4 Drivers are not available then only we should go for Type-2 Driver.

4. If no other Driver is available then only we should go for Type-1 Driver.