

## HikariCp

=====

HikariCP is a "zero-overhead" production ready JDBC connection pool.

It is the 3rd party supplied jar which helps us to improve connection pooling approach.

To use hikaricp in our applications we need to use 2 jars

- a. hikaricp jar
- b. sl4j jar

```
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;
import com.zaxxer.hikari.HikariConfig;
import com.zaxxer.hikari.HikariDataSource;
public class TestApp {

    public static void main(String[] args) throws Exception {

        String configFile = "src\\in\\lineuron\\main\\db.properties";
        HikariConfig config = new HikariConfig(configFile);

        try (HikariDataSource dataSource = new HikariDataSource(config)) {

            // Getting the connection object from connection pool
            Connection connection = dataSource.getConnection();
            Statement statement = connection.createStatement();
            ResultSet resultSet = statement.executeQuery("select
id,name,age,address from employees");
            System.out.println("ID\tNAME\tAGE\tADDRESS");
            while (resultSet.next()) {
                System.out.println(resultSet.getInt(1) + "\t" +
resultSet.getString(2) + "\t" + resultSet.getInt(3)
+ "\t" + resultSet.getString(4));
            }
        }
    }
}
```

## db.properties

=====

```
jdbcUrl=jdbc:mysql:///octbatch
dataSource.user=root
dataSource.password=root123
```

## output

ID	NAME	AGE	ADDRESS	
1	sachin	50		MI
2	dhoni	41	CSK	
3	kohli	33	RCB	
4	Gill	23		GT
7	rohith	38		MI

Note: Since hikaricp datasource is best for connection pooling, In Spring, SpringBoot by default hikaricp connection pooling is available.

javax.sql.RowSet

=====

It is an alternative to ResultSet.

We can use RowSet to handle group of records in more effective way than ResultSet.

RowSet is a child interface of ResultSet.

By default RowSet is scrollable and updatable.

By default RowSet object implements Serializable, so RowSet object can be sent over the network.

ResultSet object is by default Connected object, where as if use RowSet we can work in both the modes like

Connected/Disconnected mode.

There are 2 types of RowSet

- a. Connected RowSet.
- b. DisConnected RowSet.

Creation of different row set objects

=====

```
package in.ineuron.main;
```

```
import javax.sql.rowset.CachedRowSet;  
import javax.sql.rowset.FilteredRowSet;  
import javax.sql.rowset.JdbcRowSet;  
import javax.sql.rowset.JoinRowSet;  
import javax.sql.rowset.RowSetFactory;  
import javax.sql.rowset.RowSetProvider;  
import javax.sql.rowset.WebRowSet;
```

```
public class TestApp {
```

```
    public static void main(String[] args) throws Exception {
```

```
        RowSetFactory rsf = RowSetProvider.newFactory();  
        JdbcRowSet jrs = rsf.createJdbcRowSet();  
        CachedRowSet crs = rsf.createCachedRowSet();  
        WebRowSet wrs = rsf.createWebRowSet();  
        JoinRowSet jnrs = rsf.createJoinRowSet();  
        FilteredRowSet frs = rsf.createFilteredRowSet();
```

```
        System.out.println(jrs.getClass().getName());  
        System.out.println(crs.getClass().getName());  
        System.out.println(wrs.getClass().getName());  
        System.out.println(jnrs.getClass().getName());  
        System.out.println(frs.getClass().getName());
```

```
    }
```

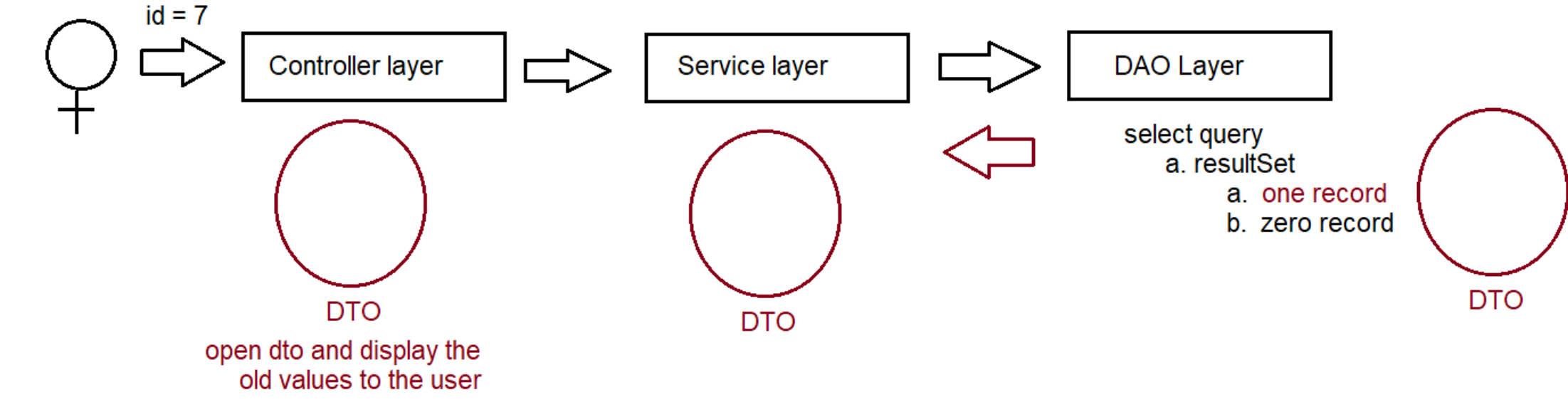
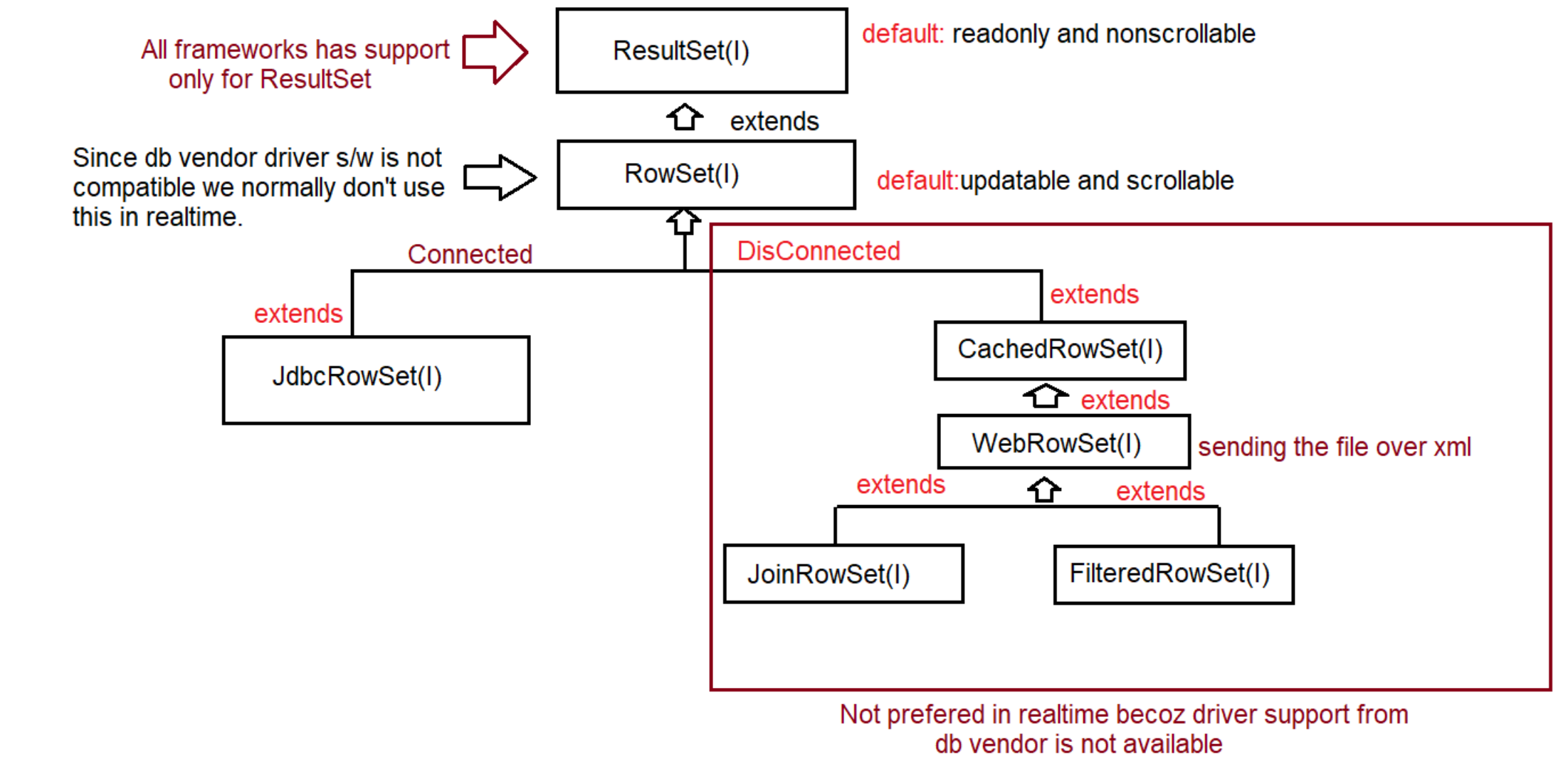
```
}
```

Output

```
com.sun.rowset.JdbcRowSetImpl  
com.sun.rowset.CachedRowSetImpl  
com.sun.rowset.WebRowSetImpl  
com.sun.rowset.JoinRowSetImpl  
com.sun.rowset.FilteredRowSetImpl
```

Note: Implementation for RowSet is provided by java vendor only, not by db vendors.





old name is sachin newname is sachintendulkar

old age is 49 newage is \_\_\_\_\_

old address is MI newAddress is IND

