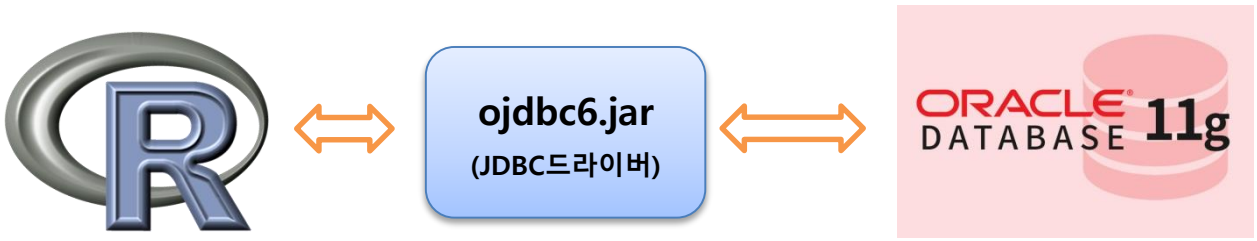
- R과 Oracle 연동



[구현 과정]

(1) 드라이버프로그램 로딩(JDBC() 함수)
(2) DBMS 서버 접속(dbConnect() 함수)
(3) 수행하려는 데이터 처리 기능에 따라서 삽입, 삭제, 수정, 추출관련 SQL 문을 전달하거나
관련 함수를 수행시켜서 데이터베이스를 사용한다.

R 언어에서 RDBMS 를 연동하여 데이터를 처리하려는 경우 RJDBC 패키지와 DBI 패키지를 사용한다.

```
install.packages("DBI"); library(DBI)
install.packages("RJDBC"); library(RJDBC)
```

RJDBC 패키지가 제공하는 함수들

JDBC
JDBCConnection
JDBCDriver
JDBCResult

DBI 패키지가 제공하는 함수들

dbBind
dbClearResult
dbColumnInfo
dbConnect
dbDataType
dbDisconnect
dbExecute
dbExistsTable
dbFetch
dbGetException
dbGetInfo
dbGetQuery
dbGetRowCount
dbGetRowsAffected
dbGetStatement
dbHasCompleted
DBIConnection
DBIDriver
DBIObject
DBIResult
dbQuoteString
dbReadTable
dbRemoveTable
dbSendQuery
dbSendStatement
dbSendUpdate
dbWithTransaction
dbWriteTable
makedbnames
rownames
SQL
sqlAppendTable
sqlCreateTable
sqlData
sqlInterpolate
Table
dbIsValid
dbListConnections
dbListFields
dbListResults
dbListTables
dbQuoteIdentifier

- DB 서버 접속
```
drv <- JDBC("oracle.jdbc.driver.OracleDriver","JDBC 드라이버압축파일패스")
conn <- dbConnect(drv, "jdbc:oracle:thin:@localhost:1521:xe", "계정","암호")
```

- DB 접속 해제
```
        dbDisconnect(conn)
```

- 테이블 리스트 추출
```
        dbListTables(conn)
```

- 테이블에 저장된 데이터 읽기

방법 1
```
        result1 <- dbReadTable(conn, "VISITOR")
```
방법 2
```
        result2 <- dbGetQuery(conn, "SELECT * FROM VISITOR")
```
방법 3
```
        rs <- dbSendQuery(conn, "SELECT * FROM VISITOR")
        ret1 <- dbFetch(rs, 1)
        ret2 <- dbFetch(rs, 1)
                :
```

- 테이블에 데이터 저장하기

방법 1
```
        dbWriteTable(conn, "book",
                data.frame(bookname=c("자바의 정석","하둡 완벽 입문","이것이 리눅스다"),
                price=c(30000, 25000, 32000)))
        )
        dbWriteTable(conn, "cars", head(cars, 3))
```
방법 2
```
        dbSendUpdate(conn, "INSERT INTO VISITOR VALUES
                                ('R 언어', sysdate, 'R 언어로 데이터를 입력해요')")
        dbSendUpdate (conn, "INSERT INTO VISITOR VALUES
                                ('하둡', sysdate, '대용량 데이터 분산저장&처리기술')")
```

- 데이터 수정
```
dbSendUpdate (conn, "INSERT INTO cars (speed, dist) VALUES (1, 1)")
dbSendUpdate (conn, "INSERT INTO cars (speed, dist) VALUES (2, 2)")
dbReadTable(conn, "CARS")
dbSendUpdate (conn, "UPDATE CARS SET DIST = DIST * 100   WHERE SPEED = 1")
dbReadTable(conn, "CARS")
dbSendUpdate(conn, "UPDATE CARS SET DIST = DIST * 3   WHERE SPEED = 1")
dbReadTable(conn, "CARS")
```

- 테이블 삭제
```
dbRemoveTable(conn, "CARS")
```

[다양한 DB 연동 예제들]

- 예제 1

```
df <- read.table("product_click.log", stringsAsFactors=F)
names(df) <- c("click_time", "pid")
df$click_time <- as.character(df$click_time)
dbWriteTable(conn, "productlog", df)
result4 <- dbReadTable(conn, "PRODUCTLOG")
```

- 예제 2

```
dbWriteTable(conn, "mtcars", mtcars)
rs <- dbSendQuery(conn, "SELECT * FROM mtcars WHERE cyl = 4")
dbFetch(rs)
dbClearResult(rs)

rs <- dbSendQuery(conn, "SELECT * FROM mtcars")
ret1 <- dbFetch(rs, 10)
ret2 <- dbFetch(rs)
dbClearResult(rs)
nrow(ret1)
nrow(ret2)
```

- Java 와 R 연동

RServe

R 바이너리 서버라고 불리우는 프로그램으로서 Java 나 다른 언어에서 R 코드를 연동할 때 필요한 기능을 서포트하는 서버이다. Java, C, C++, PHP 와 같은 다른 프로그램에서 TCP/IP 로 R 에 원격 접속, 인증, 파일전송을 가능하게 해준다. Rserve 는 최근까지도 최신 버젼이 지속적으로 업데이트되고 있다. 다음과 같이 패키지 설치 방법을 사용한다.

```
install.packages("Rserve")
```

rJava 패키지

Java 언어로 R 기술을 연동할 때 필요한 기본 API 를 담고 있는 패키지이다.

[Rserve 를 사용하기 위한 pom.xml 설정]

Java 에서는 R 을 사용하기 위해서는 Rserve 와 REngine 의 2 개의 라이브러리를 import 해야 사용할 수가 있다. 해당 라이브러리는 Rserve 홈페이지에 가서 다운로드 받아 사용할 수도 있지만, 아래와 같이 pom.xml 에 dependency 해서 사용 할 수도 있다.

```
<dependency>
    <groupId>com.github.lucarosellini.rJava</groupId>
    <artifactId>JRIEngine</artifactId>
    <version>0.9-7</version>
</dependency>
<dependency>
    <groupId>net.rforge</groupId>
    <artifactId>Rserve</artifactId>
    <version>0.6-8.1</version>
</dependency>
```

[Rserve 에 접속하는 명령 실행 – RConnection]

RConnection 은 R 에 접속 하여 핵심적인 역할을 수행하는 클래스이다. R 에 접속, 인증, 세션 종료, 파일 생성, 파일 읽기, 자료 전송, 자료 조회 등을 처리한다.

eval() - R 에 직접적인 명령을 내리고 REXP 타입으로 데이터를 반환 받는다.

assign() - R 의 변수에 REXP 또는 String 형태로 데이터를 지정하여 설정한다.

[REXP 타입]

R 과 Java 에서 서로의 자료구조와 데이터 타입을 서로 사용할 수 있도록 지원하는 데이터 모델 형의 클래스이다. 이 클래스에서 데이터 프레임과 행렬 구조로 데이터 모델을 생성 시킬 수 있다.

asBytes	Byte 일차원 배열형으로 반환하여 준다.
asDouble	double 형으로 반환하여 준다,
asDoubleMatrix	double 이차원 배열형으로 반환하여 준다.
asDoubles	double 일차원 배열형으로 반환하여 준다.
asList	RList 형으로 반환하여 준다.
asString	String 형으로 반환하여 준다.
asStrings	String 일차원 배열형으로 반환하여 준다.
length	데이터의 개수를 알수 있다.

[R 데이터 프레임을 Java 에서 사용 – RList]

Map 인터페이스를 구현하고 있는 RList 는 내부적으로 Vector 값들을 가진 리스트들이 관리하고 있다. RList 를 이용하여 데이터 프레임과 같은 자료 구조를 사용 할 수 있다.

at	Index 또는 변수명에 해당하는 열 데이터들을 REXP 객체로 반환한다.
size	리스트의 개수를 알 수 있다.

[Rserve 기동]

(1) RStudio 에서 기동시키기

```
Rserve(args="--RS-encoding utf8")
```

(2) CMD 창에서 단독으로 기동시키기(단독 기동 가능, 오류 메시지 확인 장점)

C:₩Users₩student₩Documents₩R₩win-library₩3.6₩Rserve₩libs₩x64(윈도우 10)
C:₩Program Files₩R₩R-3.6.1₩library₩Rserve₩libs₩x64(윈도우 7)의 모든 파일을
C:₩Program Files₩R₩R-xxxx₩bin₩x64 디렉토리에 복사한 후에
cmd 창을 띄우고 C:₩Program Files₩R₩R-xxxx₩bin₩x64
디렉토리에 가서 다음 명령을 수행시킨다.

```
Rserve  --RS-encoding  utf8
```

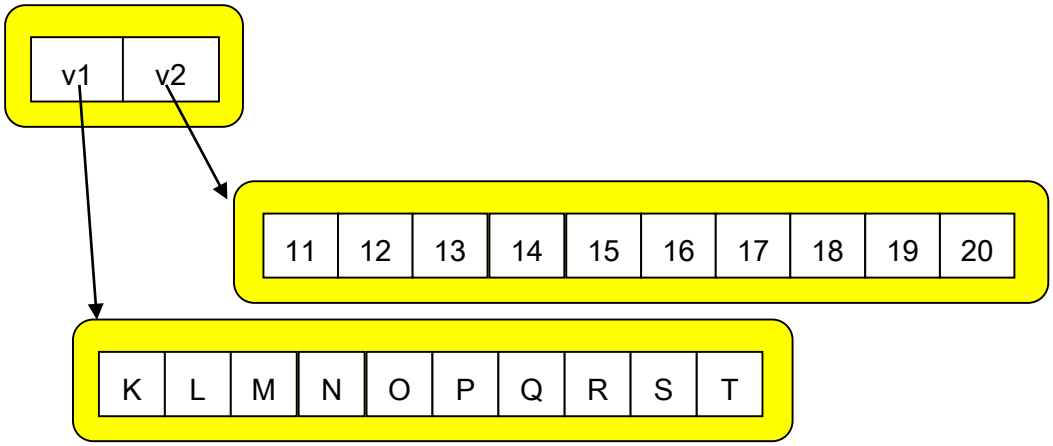
[예제 1]

```
package rtest;
import org.rosuda.REngine.REXP;
import org.rosuda.REngine.RList;
import org.rosuda.REngine.REXPMismatchException;
import org.rosuda.REngine.REngineException;
import org.rosuda.REngine.Rserve.RConnection;
```

```
import org.rosuda.REngine.Rserve.RserveException;
public class RServeExample {
	public static void getString() throws RserveException, REXPMismatchException {
		RConnection rc = new RConnection();
		REXP x = rc.eval("R.version.string");
		System.out.println("R 버전 정보 : " + x.asString());
		rc.close();
	}
	public static void getInteger() throws RserveException, REXPMismatchException {
		RConnection rc = new RConnection();
		REXP x = rc.eval("length(LETTERS)");
		System.out.println("알파벳 갯수 : " + x.asInteger());
		rc.close();
	}
	public static void getDoubles() throws RserveException, REXPMismatchException {
		RConnection rc = new RConnection();
		REXP x = rc.eval("rnorm(20)");
		double[] d = x.asDoubles();
		for (int i = 0; i < d.length; i++) {
			System.out.println(d[i]);
		}
		rc.close();
	}
	public static void getIntegers() throws REngineException, REXPMismatchException {
		RConnection rc = new RConnection();
		double[] dataX = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
		rc.assign("x", dataX); // 한글 설정시에는 설정후 iconv() 호출 필수
		rc.eval("y <- x + 10");
		int[] resultX = rc.eval("y").asIntegers();
		for (int i = 0; i < resultX.length; i++) {
			System.out.println(resultX[i]);
		}
		rc.close();
	}
	public static void getDataFrame1() throws RserveException, REXPMismatchException {
		RConnection rc = new RConnection();
		REXP x = rc.eval("d<-data.frame(LETTERS[11:20],c(11:20), stringsAsFactors=F)");
		RList list = x.asList();
		int v_size = list.size();
		int d_length = list.at(0).length();
		System.out.println("데이터(관측치)의 갯수 : " +d_length);
		System.out.println("변수의 갯수 : " +  v_size);
		int arrayRows =  v_size;
		int arrayCols =d_length;
		String[][] s = new String[arrayRows][];  //데이터프레임의 변수 갯수로 행의 크기
```

```
		for (int i = 0; i < arrayRows; i++) {
			s[i] = list.at(i).asStrings();
		}
		for (int i = 0; i < arrayRows; i++) {
			for (int j = 0; j < arrayCols; j++) {
				System.out.print(s[i][j]+"₩t");
			}
			System.out.println();
		}
		rc.close();
	}
	public static void getDataFrame2() throws RserveException, REXPMismatchException {
		RConnection rc = new RConnection();
		REXP x = rc.eval("imsi<-source('c:/RStudy/exam1/test.R'); imsi$value");
		RList list = x.asList();
		String pid = list.at("product").asString();
		System.out.print("PID : " + pid);
		String clickcount = list.at("clickcount").asString();
		System.out.println("₩tCLICKCOUNT : " +clickcount);
		rc.close();
	}
	public static void main(String[] args) throws REXPMismatchException, REngineException {
		System.out.println("------------ R 에서 버젼정보 가져오기 --------------");
		RServeExample.getString();
		System.out.println("------------ R 에서 정수 데이터 가져오기 --------------");
		RServeExample.getInteger();
		System.out.println("------------ R 에서 더블 데이터들 가져오기 -------------");
		RServeExample.getDoubles();
		System.out.println("------------  R 에서 데이터 주입 연산후 가져오기 ------");
		RServeExample.getIntegers();
		System.out.println("------------  R 에서 데이터 생성 연산후 가져오기------");
		RServeExample.getDataFrame1();
		System.out.println("------------ R 에서 정수 데이터들 가져오기 --------------");
		RServeExample.getDataFrame2();
	}
}


rc.eval("d<-data.frame(LETTERS[11:20],c(11:20), stringsAsFactors=F)");
REXP x = rc.eval("names(d) <- c('v1', 'v2')");
RList list = x.asList();
int v_size = list.size();   // 2
int d_length = list.at(0).length();   // 10
int d_length = list.at("v1").length();
```



servlet-context.xml 에 다음 내용을 반드시 추가한다.
<context:component-scan base-package="rtest" />

[R Java Oracle 예제]

```
package rtest;
import org.rosuda.REngine.RList;
import org.rosuda.REngine.Rserve.RConnection;
import org.springframework.stereotype.Repository;
@Repository
public class ROracleDB {
	public String returnDBData(int type) {
		String retStr = "";
		RConnection r = null;
		try {
			r = new RConnection();
			r.eval("library(DBI)");
			r.eval("library(RJDBC)");
			r.eval("library(rJava)");
			r.eval("drv <-
JDBC('oracle.jdbc.driver.OracleDriver','c:/unico/ojdbc6.jar')");
			r.eval("conn <- dbConnect(drv,
					'jdbc:oracle:thin:@localhost:1521:xe','work','work')");
			if( type == 1)
				r.eval("query = 'SELECT * FROM VISITOR'");
			else if (type == 2)
				r.eval("query = 'SELECT * FROM VISITOR order by name'");
			RList list = r.eval("dbGetQuery(conn,query)").asList();
			int cols = list.size();
			int rows = list.at(0).length();
			String[][]s = new String[cols][];
			for(int i=0; i<cols; i++) {
```

```
                        s[i] = list.at(i).asStrings();
                }
                for(int j=0; j<rows; j++) {
                        for(int i=0; i<cols; i++) {
                                retStr += (s[i][j])+"";
                        }
                        retStr += "</br>";
                }
        } catch(Exception e) {
                System.out.println(e);
                retStr = "오류 발생!!";
        } finally {
                r.close();
        }
        return retStr;
    }
    public String insertDBData(String name, String content) {
        String retStr = "";
        RConnection r = null;
        System.out.println("---->"+name);
        System.out.println("---->"+content);
        try {
                r = new RConnection();
                r.eval("library(DBI)");
                r.eval("library(RJDBC)");
                r.eval("library(rJava)");
                r.eval("drv <-
JDBC('oracle.jdbc.driver.OracleDriver','c:/unico/ojdbc6.jar')");
                r.eval("conn <- dbConnect(drv,
                                'jdbc:oracle:thin:@localhost:1521:xe','work','work')");
                r.eval("insertSQL <-
                        'INSERT INTO visitor VALUES ("+name+",sysdate,"+content+")'");
                r.eval("dbSendUpdate (conn, insertSQL)");
                retStr = r.eval("'정상적으로 저장되었어요..'").asString();
        } catch(Exception e) {
                System.out.println(e);
                retStr = "오류 발생!!";
        } finally {
                r.close();
        }
        return retStr;
    }
}
```