제10장
JDBC연동과 DB연동 객체들

1. JDBC란?

JAVA 프로그램에서 SQL문을 실행하여 데이터를 관리하기 위한 JAVA API입니다. JDBC의 특징은 다양한 데이터 베이스에 대해서 별도의 프로그램을 만들 필요 없이, 해당 데이터 베이스의 JDBC를 이용하면 하나의 프로그램으로 데이터 베이스를 관리할 수 있다.
우리는 MySQL을 사용하므로, MySQL용 JDBC를 사용하며, 오라클 홈페이지에서 다운받아서, 이클립스에 해당 클래스 파일을 복사 하면 된다. 타 DB도 역시 JDBC를 다 제공하므로, DB가 바뀌어도 해당 DB의 JDBC만 설정해주면 된다.

mysql 드라이버를 사용하기 위한 mysql-connector-java_버전.jar 파일 복사

* 참고
JDBC : Java Data Base Connectivity, 자바전용
ODBC : Open Data Base Connectivity, MS사 개발

2. JDBC를 사용한 DB연동 순서

2. JDBC를 사용한 **JSP**와 데이터베이스의 연동

❶ java.sql.* 패키지 임포트,
❷ JDBC 드라이버 로딩,
❸ 데이터베이스 접속을 위한 Connection 객체 생성,
❹ 쿼리문을 실행하기 위한 Statement/PreparedStatement/CallableStatement객체 생성,
❺ 쿼리 실행,
❻ 쿼리 실행의 결과 값(int, ResultSet) 사용,
❼ 사용된 객체(ResultSet, Statement/PreparedStatement/CallableStatement, Connection) 종료

1. JDBC 드라이버 로드	2. 데이터베이스 연결	3. SQL문 실행	4. 데이터베이스 연결 해제

위의 **1~4**번까지의 순서를 반드시 기억을 하자.
그래야 자바와 **DB**를 연동함에 혼돈이 없다.

3. JDBC 드라이버 로딩 및 DBMS 접속

3.1 JDBC 드라이버 로딩하기

JDBC 드라이버 로딩 단계에서는 드라이버 인터페이스를 구현하는 작업

Class.forName( ) 메소드를 이용하여 JDBC 드라이버를 로딩

```
Class.forName(String className);
```

JDBC 드라이버가 로딩되면 자동으로 객체가 생성되고 DriverManager 클래스에 등록

[MySQL 드라이버 로딩 예]

```
<%
    try{
       Class.forName("com.mysql.jdbc.Driver");
    }catch(SQLException ex){
       //예외 발생 처리
    }
%>
```

JDBC 드라이버 로딩은 프로그램 수행 시 한 번만 필요

3. JDBC 드라이버 로딩 및 DBMS 접속

3.2 Connection 객체 생성하기

- JDBC 드라이버에서 데이터베이스와 연결된 커넥션을 가져오기 위해 DriverManager 클래스의 getConnection( ) 메소드를 사용
- DriverManager 클래스로 Connection 객체를 생성할 때 JDBC 드라이버를 검색하고, 검색된 드라이버를 이용하여 Connection 객체를 생성한 후 이를 반환

```
static Connection getConnection(String url)
static Connection getConnection(String url, String user, String password)
static Connection getConnection(String url, Properties info)
```

[Connection 객체 생성 예: getConnection(String url) 메소드 사용]

```
<%
    Connection conn = null;
    try {
       Class.forName("com.mysql.jdbc.Driver");
       conn = DriverManager.getConnection(
       "jdbc:mysql://localhost:3306/JSPBookDB?user=root&password=12341");
    } catch (SQLException ex) {
       //예외 발생 처리
    }
%>
```

3. JDBC 드라이버 로딩 및 DBMS 접속

3.3.1 Statement 인터페이스 살펴보기

- 정적인 쿼리에 사용
- 하나의 쿼리를 사용하고 나면 더는 사용할 수 없음
- 하나의 쿼리를 끝내면 close( )를 사용하여 객체를 즉시 해제해야 함
- close( )를 사용하여 객체를 즉시 해제하지 않으면 무시할 수 없는 공간이 필요하며, 페이지가 다른 작업을 수행하는 동안 멈추지 않기 때문
- 복잡하지 않은 간단한 쿼리문을 사용하는 경우에 좋음

메소드	반환 유형	설명
executeQuery(String sql)	ResultSet	SELECT 문을 실행할 때 사용합니다(ResultSet 객체 반환).
executeUpdate(String sql)	int	삽입, 수정, 삭제와 관련된 SQL 문 실행에 사용합니다.
close( ):	void	Statement 객체를 반환할 때 사용합니다.

3. JDBC 드라이버 로딩 및 DBMS 접속

3.3.2 Statement 인터페이스 살펴보기

- executeQuery() 메소드로 데이터 조회하기

```
ResultSet executeQuery(String sql) throws SQLException
```

[executeQuery() 메소드 사용 예: SELECT 쿼리문]

```
<%
    Connection conn = null;
    …(생략)…
    Statement stmt = conn.createStatement();
    String sql = "SELECT * FROM Member WHERE id ='1'";
    ResultSet rs = stmt.executeQuery(sql);
    stmt.close();
%>
```

3. JDBC 드라이버 로딩 및 DBMS 접속

3.3.2 Statement 인터페이스 살펴보기

- executeUpdate() 메소드로 데이터 삽입, 수정, 삭제하기

  executeUpdate( ) 메소드는 INSERT, UPDATE, SELECT 쿼리문을 통해 데이터를 삽입, 수정, 삭제하는 데 사용

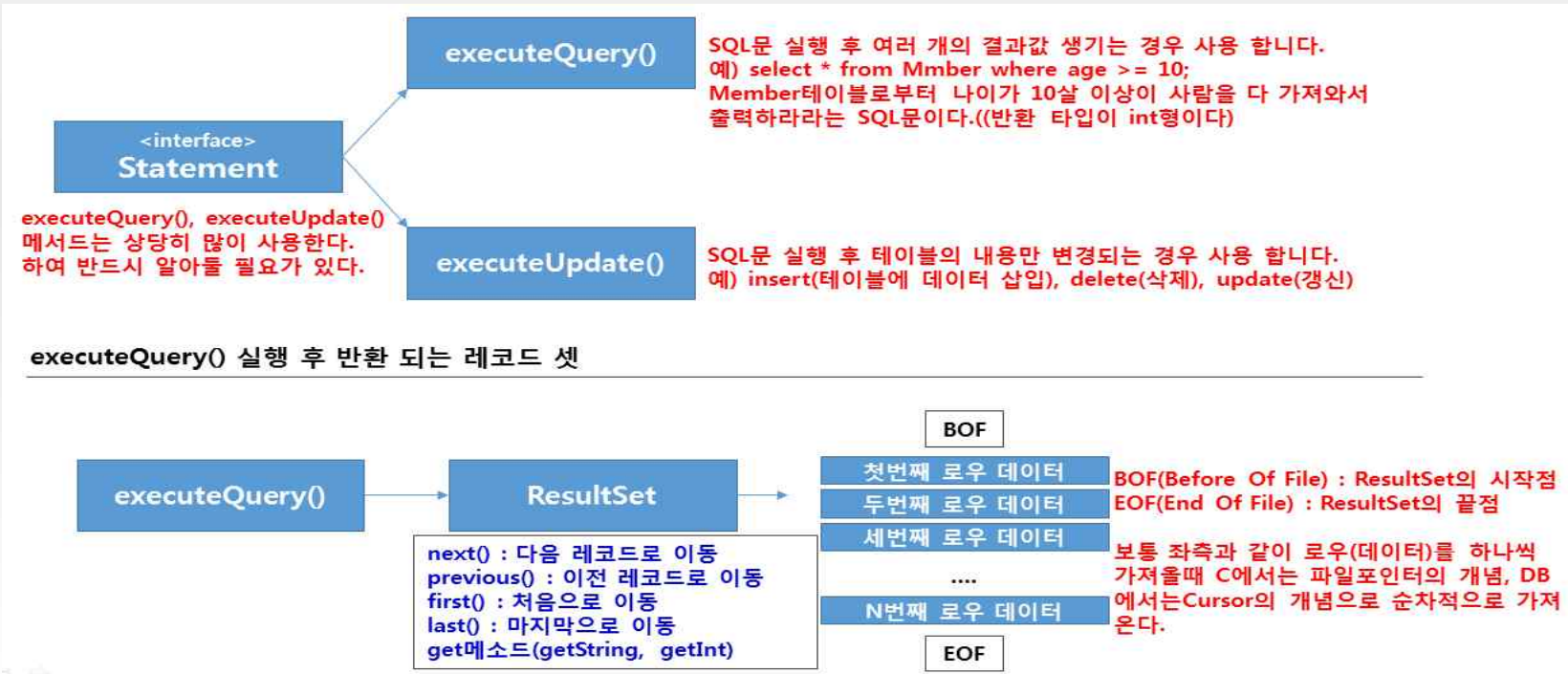
```
int executeUpdate(String sql) throws SQLException
```

[executeUpdate() 메소드 사용 예(삽입): INSERT 쿼리문]

```
<%
    Connection conn = null;
    …(생략)…
    Statement stmt = conn.createStatement();
    String sql = "INSERT INTO Member(id, name, passwd) VALUES ('1', '홍길순',
    '1234')";
    int rs = stmt.executeUpdate(sql);
%>
```

3. JDBC 드라이버 로딩 및 DBMS 접속

3.3.2 Statement 인터페이스로 SQL문 실행



3. JDBC 드라이버 로딩 및 DBMS 접속

3.4 데이터베이스 연결 닫기

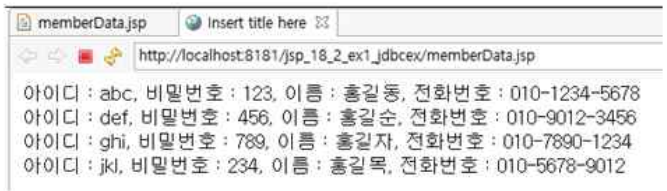
- 데이터베이스 연결이 더 이상 필요하지 않으면 데이터베이스와 JDBC 리소스가 자동으로 닫힐 때까지 대기하는 것이 아니라 close( ) 메소드로 생성한 Connection 객체를 해제
- 일반적으로 데이터베이스 리소스를 사용하지 않기 위해 사용을 끝내자마자 리소스를 해제하는 것이 좋음

```
<%
    Connection conn = null;
    try{
       //JDBC 드라이버 로딩
       //Connection 객체 생성
    } catch(SQLException e){
       //예외 발생 처리
    } finally{
       if (conn != null) conn.close();
    }
%>
```

4. JDBC를 사용한 DB연동 예제

JDBC예제

```
String driver = "com.mysql.jdbc.Driver";  //mysql드라이버 명시
//mysql접속 위치(jdbc:mysql: -> 프로토콜, //localhost
//-> 도메인, :3306 ->포트번호, sqldb->접속 DB명
String url = "jdbc:mysql://localhost:3306/sqldb";
String id = "root";      //접속 아이디
String pw = "1234";      //접속 비밀번호
String query = "select * from member order by name";    //쿼리문 작성
```




```
<%
   try{

      Class.forName(driver);
      connection = DriverManager.getConnection(url, uid, upw);
      statement = connection.createStatement();
      resultSet = statement.executeQuery(query);

      while(resultSet.next()){
         String id = resultSet.getString("id");
         String pw = resultSet.getString("pw");
         String name = resultSet.getString("name");
         String phone = resultSet.getString("phone");

         out.println("아이디 : " + id + ", 비밀번호 : " + pw + ", 이름 : " + name + ", 전화번호 : " + phone + "<br />");
      }
```

** 아래 부분은 암기하도록 하자.

1. Driver 로드
2. 드라이버 매니저 클래스로부터 Connection객체 얻기
3. 커넥션 객체로부터 쿼리를 실행할 수 있게 하는 Statement객체 얻기
4. 쿼리 후, 결과를 얻는 ResultSet객체 얻기
5. 반복문으로 결과 출력

감사합니다.