postgreSQL의 JDBC 활용하기

주효식 nogadax@kldp.org 2000년 6월 15일

이 문서는 "Apache-JServ 에 postgreSQL 연동하기"라는 제목의 문서를 대폭 수정, 보완한 것이다. 또한 서버 클라이언트에서 JDBC를 설정하는 사항도 추가하였다. 이 문서는 포스트그레스의 JDBC를 활용하는 방법을 크게 두가지로 나누어 설명한다. 첫 번 째는 postgreSQL을 DB 서버로 하고 윈도우(Window 98)를 클라이언트로 할때 JDBC의 설정사항에 대해 설명한다. 두 번째는 Jserv를 추가한 Apache에 DB 서버로서 postgreSQL 을 연동하기 위해 JDBC의 설정 사항등에 대해 설명한다.

목 차

제 1 절 준비 사항

설정 및 테스트에 앞서 기본적으로 준비되어야 할 사항이 있다. 이에 대한 사항은 *http://www.kldp.org*

<http://www.kldp.org> 를 참조하거나 기타 문서를 참조하기 바란다. 다음은 미리 준비되어 야 할 사항이다.

- 아파치
- JServ
- JDK1.1.x 이상
- JSDK2.0 이상
- postgreSQL
- Window 98 (클라이언트)

이미 테스트를 마친 필자의 환경은 다음과 같다.

- 액셀 리눅스 6.1
- 아파치 1.3.9
- JServ 1.0
- JDK 1.2.2
- JSDK 2.0
- postgresql 6.5.3 및 postgresql 7.0.x
- Window 98 (클라이언트)

기타 참고를 위한 사이트는 다음과 같다.

- *http://apache.kr.net* <http://www.apache.kr.net>
- *http://www.apache.org* <http://www.apache.org>
- *http://www.postgresql.org* <http://www.postgresql.org>
- *http://www.blackdown.org* <http://www.blackdown.org>
- *http://java.apache.org* <http://java.apache.org>
- *http://java.sun.com* <http://java.sun.com>
- *http://database.sarang.net* <http://database.sarang.net>

기타 약간의 포스트그레스 문서와 Apache-JServ, 포스트그레스 연동 관련의 소스 예제가 필자의 홈페이지에 (*http://user.chollian.net/˜nogadax* <http://user.chollian.net/~nogadax>) 있으니 참조하기 바란다.

제 2 절 포스트그레스 JDBC 드라이버 만들기

포스트그레스는 **Type 4**의 JDBC 드라이버를 제공한다. Type 4는 순수 자바로 작성되었고 데이터베이스 자체의 통신 프로토콜로 통신한다. 이는 클라이언트측이 JDBC를 통해 DBMS 서버와의 직접적인 통신이 가능함을 의미한다. 네트워크상에서 JDBC 드라이버를 통해 간편 하게 포스트그레스 DB 서버와 연결할 수 있으며 이를 응용하려는 개발자는 DB 의 기능에 만 집중하며 작업을 할 수 있다.

또한 포스트그레스 JDBC 드라이버는 플랫폼에 비종속적이므로 한번 컴파일된 드라이버는 어떤 플랫폼에서도 사용 가능하다. 예를 들어, 컴파일 되어 생성된 포스트그레스 JDBC 드 라이버는 윈도우나 리눅스에서 어떠한 수정없이 바로 사용 가능하다.

2.1 JDBC 다운받기

포스트그레스를 직접 설치하지 않아 포스트그레스의 소스가 없을 경우 포스트그레스 사이트 (*http://www.postgresql.org* `<http://www.postgresql.org>`) 에서 RPM 형태의 JDBC 드라이버를 다운받을 수 있다. 그외에 *http://www.retep.org.uk* `<http://www.retep.org.uk>` 에서 다운 받을 수 있다.

다운받을 때 주의할 점이 있는데 만약 "JDK1.1.x" 를 가지고 있다면 "JDBC 1" 드라이버를 다운받아야 하고 "JDK1.2" 이상을 가지고 있다면 "JDBC 2" 드라이버를 다운받으면 된다.

2.2 JDBC 만들기

JDBC 드라이버가 없다면 이를 만들어야 한다. 만약, 포스트그레스의 소스를 다운받아 직접 설치를 한 경우이면 포스트그레스 소스 디렉토리내의 interfaces 디렉토리에 JDBC 드라이버 소스가 있어서 컴파일을 하면 된다.

JDBC 드라이버의 컴파일시 "JDK1.1.x" 하에서 JDBC 드라이버를 만들면 "JDBC 1" 드라이 버이다. JDK1.2 버젼에서 만들어진 JDBC 드라이버는 "JDBC 2" 드라이버이다.

이 문서에서는 포스트그레스의 소스 디렉토리는 "/usr/local/src/pgsql_버젼" 이라 가정하고 포스트그레스가 컴파일되어 설치될 디렉토리를 "/usr/local/pgsql" 이라고 가정한다. 기타, 포 스트그레스 "6.5.x" 버전대의 컴파일 방법과 포스트그레스 "7.0.x" 버전대의 JDBC 드라이버 의 컴파일 방법이 다르므로 주의하여야 한다. 이제 JDBC 드라이버를 만들어보자.

2.2.1 postgreSQL 6.5.x 드라이버 만들기

postgreSQL 6.5.x 의 경우는 다음과 같다.

/usr/local/src/pgsql6.5.3/src/interfaces/jdbc]$**make**

단지 위의 디렉토리로 가서 "make"만 해주면 된다. "make" 완료 후에 컴파일 된 드라이버 를 확인해 보자. 드라이버의 이름은 "postgresql.jar" 이다. 이제는 "/usr/local/pgsql" 에 jdbc 라는 디렉토리를 만들고 "postgresql.jar" 를 복사하자. JDBC 드라이버를 저장할 디렉토리는 자신이 원하는 대로 바꾸어도 좋다.

/usr/local/pgsql]$ mkdir jdbc

/usr/local/pgsql]$ cp ../src/pgsql6.5.3/src/interfaces/postgresql.jar ./jdbc/

2.2.2 postgreSQL 7.0.x 드라이버 만들기

"postgreSQL 7.0.x" 의 경우는 "6.5.x" 와는 좀 다르다. 만약, "6.5.x" 처럼 "make" 만 입력 하면 메시지가 나오는데 그 내용은 "JDK1.1.x" 이면 "make jdbc1 jar"을 입력하고 "JDK1.2.x" 이면 "make jdbc2 jar"를 입력하라는 메시지가 나온다. 여기서는 "JDK1.2.x" 로 가정하고 컴파일 해보자.

/usr/local/src/pgsql7.0.2/src/interfaces/jdbc]$**make jdbc2 jar**

이제 컴파일 된 드라이버를 확인해 보자. 드라이버의 이름은 "postgresql.jar" 이다. 생성된 JDBC 드라이버는 "/usr/local/pgsql" 에 jdbc 라는 디렉토리를 만들고 JDBC 드라이버를 복 사해 놓으면 된다.

/usr/local/pgsql]$ mkdir jdbc

/usr/local/pgsql]$ cp ../src/pgsql7.0.2/src/interfaces/postgresql.jar ./jdbc/

2.3 JDBC 드라이버 테스트하기

리눅스에서 컴파일되어 생성된 JDBC 드라이버를 테스트하려고 하는 경우 "JDK"를 위해 CLASSPATH 를 설정하여야 한다. CLASSPATH의 설정 예는 다음과 같다.

export CLASSPATH="$CLASSPATH:/usr/local/pgsql/jdbc/postgresql.jar:."

위의 export 명령을 일반 쉘상에서 실행하여도 되나 이는 일회성이므로 매번 부팅 때마다 export 명령을 실행해주어야 한다. 일회성이 아닌 영구적으로 설정하기 위해 리눅스의 부팅 시 로긴 관련 설정 파일인 "**/etc/profile**" 이나 "/root/.bash_profile" 등에 위의 CLASSPATH 설정 예를 추가하면 된다.

설정이 끝나면 부팅 후 설정 사항 및 JDBC 드라이버의 확인을 위해 테스트를 해보자. JDBC 드라이버의 테스트를 위해 "**/usr/local/src/pgsql653/src/interfaces/jdbc/example**" 디렉 토리의 예제 소스를 컴파일해본다. 다음은 테스트를 위해 example 디렉토리의 basic.java를 컴파일하는 예이다.

/usr/local/src/pgsql653/src/interfaces/jdbc/example]$ javac basic.java

제 3 절 서버 클라이언트 (윈도우) 설정

리눅스에 포스트그레스를 설치한 후 포스트그레스를 DB서버로 하고 클라이언트측인 윈도우 에서 JDBC를 통해 포스트그레스를 이용하려 한다고 가정하자. 기타 윈도우의 IP 주소는 210.110.144.169 이라 가정한다.

JDBC 드라이버의 설정 순서는 다음과 같다.

1. 포스트그레스 JDBC 드라이버를 윈도우의 적당한 디렉토리에 복사한다.
2. 윈도우의 AUTOEXEC.BAT 파일에 JDBC 드라이버 경로를 CLASSPATH에 등록한다.

   아래는 AUTOEXEC.BAT 파일에서의 설정예이다.

   set classpath=.;c:\postgres\jdbc_drv\postgresql.jar
3. 윈도우를 리부팅한다.
4. 윈도우에서 DB서버로의 원할한 접근을 허용하기 위해 리눅스에 설치된 포스트그레스의 "pg_hba.conf" 파일을 수정한다.

   만약 포스트그레스의 경로가 "/usr/local/pgsql" 이면 pg_hba.conf 파일은 "**/usr/local/pgsql/data**" 디렉토리에 있다.

   아래는 설정 예로서 자신에 환경에 맞게 수정한 다음 pg_hba.conf 파일의 마지막 라인에 추 가하면 된다.

   host all 210.110.144.169 255.255.255.255 trust

위의 예에 대한 설명은 다음과 같다.

host 는 예약어로서 Internet Domain 방식을 의미한다. **all** 은 모든 데이터베이스를 의미하 며 만약 특정 DB로 설정하고자 한다면 DB 이름을 적으면 된다. 예를 들어 접근하려는 데 이터베이스명이 "nogadax" 이면 위의 예에서 "all" 대신 "nogadax"를 적으면 된다. **210.110.144.169**는 포스트그레스에 접근하려는 클라이언트의 IP주소로서 여기서는 윈도우의 IP 주소이다. **255.255.255.255** 는 Net Mask 이다. **trust** 는 클라이언트의 접근에 대한 인증 메소드로서 trust는 별다른 인증 절차없이 access 가 가능하다. 만약 **crypt** 이면 ID 와 패 스워드가 필요하다. 이에 대한 자세한 내용은 "pg_hba.conf" 설명을 참조바라며 이에 대한 것은 *http://www.kldp.org* `<http://www.kldp.org>` 에 있으니 참조하기 바란다.

테스트

테스트를 위해 포스트그레스의 JDBC 예제 프로그램 소스를 윈도우에 적당히 복사한 후 컴파일 한다. 컴파일 예는 다음과 같다.

javac Basic.java

제 4 절 Apache-Jserv에 연동하기

이 부분을 시작하기 전에 JDK를 위한 **CLASSPATH**가 먼저 설정되어 있어야 한다. 이를 위해 앞에서 설명한 "JDBC 드라이버 테스트하기"를 참조하기바란다.

Apache-JServ 가 이미 설치되어 있다면 JServ 환경파일인 "**jserv.properties**" 가 있을 것이 다. 이 화일내에 postgresql.jar의 경로를 포함시켜주면 된다. 필자의 Apache-JServ의 환경 파일 구성은 다음과 같다.

```
/usr/local/apache/conf]$ ls
   access.conf
   httpd.conf
   srm.conf
   jserv.conf
   jserv.properties
   servlets.properties
```

포스트그레스의 JDBC 드라이버가 "jserv.proeprties" 에 등록이 되면 Apache-JServ 를 위한 환경 설정은 끝이다. "jserv.properties" 에 다음처럼 추가해주면 된다.

wrapper.classpath=/usr/local/pgsql/jdbc/postgresql.jar

위처럼 추가를 하였다면 Apache-JServ 의 환경설정 부분은 끝났다.

이제는 "서버 클라이언트 (윈도우) 설정" 부분에서 언급되었던 "pg_hba.conf" 파일을 수정하 면 된다. 여기서는 Apache 가 설치된 곳의 IP 주소를 210.110.144.168 라고 가정하자. 또한, 포스트그레스의 경로가 "/usr/local/pgsql" 이면 pg_hba.conf 파일은 "/usr/local/pgsql/data" 디렉토리에 있다.

아래는 설정 예로서 자신에 환경에 맞게 수정한 다음 pg_hba.conf 파일의 마지막 라인에 추 가하면 된다.

```
host  all  210.110.144.168  255.255.255.255  trust
```

위의 예에 대한 설명은 다음과 같다.

host 는 예약어로서 Internet Domain 방식을 의미한다. Internet Domain 은 프로세스의 통 신이 네트워크간의 프로토콜을 통해 이루어지게 하는 메카니즘으로서 프르토콜을 통해 DB 서버에 접근하려는 Host 의 접근을 제어한다. **all** 은 모든 데이터베이스를 의미하며 만약 특 정 DB로 설정하고자 한다면 DB 이름을 적으면 된다. 예를 들어 접근하려는 데이터베이스 명이 "nogadax" 이면 위의 예에서 "all" 대신 "nogadax"를 적으면 된다. "**210.110.144.168**" 는 포스트그레스에 접근하려는 클라이언트의 IP주소이다. 여기서는 Apache-Jserv가 설치된 곳의 IP 주소이다. **255.255.255.255** 는 Net Mask 이다. **trust** 는 클라이언트의 접근에 대한 인증 메소드로서 trust는 별다른 인증 절차없이 access 가 가능하다. 만약 crypt 이면 ID 와 패스워드가 필요하며 더욱 자세한 내용은 pg_hba.conf 파일을 설명한 문서를 참조하기 바란다.

제 5 절 포스트그레스 JDBC 기초

5.1 JDBC를 위한 준비사항

JDBC 드라이버로 포스트그레스를 응용한 프로그램을 테스트하기 전에 포스트그레스 디몬 이 활성화 되어 있어야 하며 데몬은 다음의 옵션(i)에 의해 기동되어져야 한다.

postmaster -i &

이것은 네트워크를 통해 연결하려는 JDBC를 위해 필수 사항이다. 또한 pg_hba.conf 파일을 확인하도록 한다. 만약, 일반적인 IP주소가 아닌 **localhost(127.0.0.1)**을 사용하려한다면 이 주 소(127.0.0.1)에 대하여 pg_hba.conf 에 설정하여야 한다. 다음은 localhost 설정 예이다.

```
host  all  127.0.0.1  255.255.255.255  trust
```

5.2 드라이버 로딩

JDBC 드라이버의 로딩 방법은 **Class.forName()** 를 사용하면 된다. 주의할 점이 있는데 포 스트그레스 6.5.x 버전과 7.0.x 버전대에서 약간의 차이가 생겼다. 필자가 테스트를 목적으 로 이전에 만들어 놓았던 프로그램으로 7.0.x 버전대의 JDBC 드라이버를 테스트를 해본 결 과 에러가 발생하였다. 에러를 해결하기 위해 7.0.x 버전의 문서를 찾아본 결과 아무런 변동 이 없었지만 example 디렉토리의 예제 소스를 보고 원인을 발견하였다.

포스트그레스 6.5.x 버전의 JDBC 드라이버는 Class.forName(”**postgresql.jar**”); 하면 되지만 7.0.x 버전의 JDBC 드라이버는 Class.forName(”**org.postgresql.jar**”);를 하여야 한다. 이러한 차이를 주의하기 바란다. 다음은 포스트그레스 버전대별로 Class.forName()를 사용한 예이 다. 기타 예외 처리를 위해 **ClassNotFoundException**을 사용한다.

5.2.1 postgresql 6.5.X 에서의 Class.forName() 의 예

```
public void  test()
throws ClassNotFoundException, FileNotFoundException,
       IOException, SQLException
{
   String url = "jdbc:postgresql:nogadax";
   String usr = "postgres";
   String pwd = "";

   Class.forName("postgresql.Driver");

   Connection db = DriverManager.getConnection(url, usr, pwd);
   ........
   ........
}
```

5.2.2 postgresql 7.0.X 에서의 Class.forName() 의 예

```
public void  test()
throws ClassNotFoundException, FileNotFoundException,
       IOException, SQLException
{
   String url = "jdbc:postgresql:nogadax";
   String usr = "postgres";
   String pwd = "";

   Class.forName("org.postgresql.Driver");

   Connection db = DriverManager.getConnection(url, usr, pwd);
   ........
   ........
}
```

5.3 Connection to the Database

포스트그레스의 데이터베이스로의 연결을 위한 예는 다음과 같다.

```
jdbc:postgresql:database_name
jdbc:postgresql://host/database_name
```

```
jdbc:postgresql://host:port/database_name
```

host

포스트그레스 DB 서버의 호스트 네임이다. IP를 주어도 된다. 만약 이 부분이 생략되면 디폴트는 localhost(127.0.0.1) 이다.

port

포스트그레스의 디폴트 포트는 5432 이며 생략 가능하다.

database_name

데이타베이스명

5.3.1 기타 사항

컨넥트하기 위해 JDBC 로부터 Connection 인스턴스를 얻어야 한다. 이를 위해 DriverManager.getConnection() 메소드를 사용해야 한다.

Connection db = DriverManager.getConnection(url,user,pwd);

기타 포스트그레스는 하나의 레코드의 크기가**8K 바이트**로 제한되어 있다. 만약 하나의 레 코드가 8K 바이트를 넘어야 한다면 "**Large Object**" 라는 방법을 사용하여야 한다. 이에 대 해서는 포스트그레스 프로그래머 가이드의 JDBC 부분을 참조하기 바란다. 또한, JDBC 를 위한 나머지 자세한 부분은 일반 JDBC 문서들을 참조하기 바란다.

5.4 Jserv 테스트용 소스

다음은 Apache-Jserv 와 postgreSQL이 연동되었을 때 테스트를 위한 소스이다. 다음의 소스는 포스트그레스의 예제 소스로서 단지 servlet 로 바꾸어 web 상에서 실행이 되게끔 해준 것이다.

다음 예제를 컴파일하고 실행하기에 앞서 포스트그레스에 데이타베이스를 만들어야 한다. 여기서는 데이타베이스 이름이 nogadax 이다. 이를 적절히 바꾸기 바란다. 또한 포스트그레 스의 유저명은 postgres 이고 패스워드는 없다. 이곳의 소스에서의 url, usr, pwd 의 설정은 다음과 같다. 이를 참조하여 자신의 환경에 맞게 바꾸기 바란다.

```
String url = "jdbc:postgresql:nogadax";
String usr = "postgres";
String pwd = "";
```

필자의 테스트 환경은 한대의 PC 에 포스트그레스와 아파치가 같이 설치되어 있다. 만약 테스트하려고 하는 환경이 DB서버와 WEB서버가 서로 다른 곳에 설치되어 있다면, 즉, 3-Tier 환경이면 URL 은 위와는 다르게 되어져야 한다.

자신의 환경을 적용하기 위해 다음의 예를 참조하기 바란다. 참고로 IP 주소 210.110.144.162 는 포스트그레스 DB 서버의 IP주소이며 포트 번호가 생략되었으므로 디폴트 번호 "5432"로 자동 셋팅된 상태이다.

String url= "jdbc:postgresql://210.110.144.162/nogadax";

기타 Class.forName()을 버전에 따라 적절히 수정하기 바란다. 아래의 테스트용의 소스는 포스트그레스 6.5.3 의 JDBC 드라이버를 기준으로 작성되었으므로 Class.forName()이 아래 와 같다.

Class.forName("postgresql.Driver");

그러므로 7.0.x 의 JDBC 드라이버에서 테스트를 하려면 "**org.postgresql.Driver**" 로 수정하 면 된다.

또한 컴파일된 서블릿은 리포지토리에 있어야 한다. 리포지터리는 서블릿을 위한 논리적인 디렉토리이다. 이것은 zone 와 연결되어지는데 zone은 URL에 첨가되어진다. 만약 zone 의 이름이 servlet 이고 아래의 예

제소스를 컴파일 후 지정된 리포지터리에 있다면 실행을 위해 브라우저의 URL 창에 다음처럼 입력해주면 되는데 자신에게 맞게 바꾸어주면 된다.

http://localhost/servlet/Hello

혹은

http://210.110.144.162/servlet/Hello

혹은

http://nogadax.tit.ac.kr/servlet/Hello

다음은 "Hello.java" 라는 테스트용의 소스이다.

```
-------------------------------
import java.io.*;
import javax.servlet.*;
import java.sql.*;
import java.text.*;
import javax.servlet.http.*;

public class Hello extends HttpServlet
{
  Connection db;
  Statement  st;
  PrintWriter out;

  public void doGet (HttpServletRequest request,
                    HttpServletResponse response)
  throws ServletException, IOException
  {
   String title = "Example Apache JServ Servlet";
   response.setContentType("text/html");
   out = response.getWriter();
   out.println("<HTML><HEAD><TITLE>");
   out.println(title);
   out.println("</TITLE></HEAD><BODY bgcolor=\"#FFFFFF\">");
   out.println("<H1>" + title + "</H1>");
   out.println("<H2>Congratulations, Apache JServ is working!</H2>");
   try {
        test();
       } catch(Exception ex){
          out.println("<h2>DB error!</h2>");
         }
   out.println("</BODY></HTML>");
   out.close();
   }

   public void cleanup()
   {
     try {
         st.executeUpdate("drop table basic");
       } catch(Exception ex) {  }
   }

   public void doexample() throws SQLException
   {
```

```
    out.println("<br><h2>Running tests:</h2>");
    st.executeUpdate("create table basic (a int2, b int2)");
    st.executeUpdate("insert into basic values (1,1)");
    st.executeUpdate("insert into basic values (2,1)");
    st.executeUpdate("insert into basic values (3,1)");
    st.executeUpdate("update basic set b=8");
    out.println("<h2>Updated "+st.getUpdateCount()+" rows</h2>");
    PreparedStatement ps = db.prepareStatement("insert into
                                           basic values (?,?)");
    for(int i=2;i<5;i++) {
           ps.setInt(1,4);            // "column a" = 5
           ps.setInt(2,i);            // "column b" = i
           ps.executeUpdate();
     }
    ps.close();
    out.println("<h2>performing a query</h2>");
    ResultSet rs = st.executeQuery("select a, b from basic");
    if(rs!=null) {
        while(rs.next()) {
              int a = rs.getInt("a");
              int b = rs.getInt(2);
              out.println("<h2>  a="+a+" b="+b+"</h2>");
         }
        rs.close();
    }
  }

  public void  test()
  throws ClassNotFoundException, FileNotFoundException,
                              IOException,SQLException
  {
   String url = "jdbc:postgresql:nogadax";
   String usr = "postgres";
   String pwd = "";
   Class.forName("postgresql.Driver");
   db = DriverManager.getConnection(url, usr, pwd);
   out.println("<h2>Connecting to Database URL = " + url +"</h2>");
   out.println("<h2>Connected...Now creating a statement</h2>");
   st = db.createStatement();
   cleanup();
   doexample();
   cleanup();
   out.println("<h2>Now closing the connection</h2>");
   st.close();
   db.close();
  }//end test
}
```

실행 결과

실행 결과는 다음과 같다. 아래에서 보여지는 URL이 다를 수 있다.

```
Example Apache JServ Servlet
Congratulations, Apache JServ is working!
Connecting to Database URL = jdbc:postgresql:nogadax
```

```
Connected...Now creating a statement

Running tests:

Updated 3 rows
performing a query
a=1 b=8
a=2 b=8
a=3 b=8
a=4 b=2
a=4 b=3
a=4 b=4

Now closing the connection
```

제 6 절 포스트그레스 유저 만들기

JDBC 로 연결을 할 때 새로운 유저명과 패스워드를 주고 싶을 때가 있을 것이다. 이를 위해 포스트그레스의 유저와 유저에 대한 패스워드를 만드는 방법을 간단히 소개 하겠다.

먼저 포스트그레스의 슈퍼 계정(DBA)를 "postgres" 라고 가정하자.

여기서는 데이터베이스의 새로운 유저를 생성하고 그 유저의 데이터베이스를 생성할 것이다. 순서는 다음과 같다.

1. postgres 계정에 패스워드를 생성한다. 이를 위해 포스트그레스의 특별한 데이터베이스인 "**template1**" 에 접속한다. 패스워드를 "123qwe" 라고 가정한다.

```
postgres]$ psql template1
.......
template1=>alter user postgres with password '123qwe';
ALTER USER
template1=>\q
postgres]$
```

2. "pg_hba.conf" 파일을 수정한다. 만약, DB 서버로 접속하려는 자신의 Host 의 IP 가 "210.110.144.169" 이면 다음처럼 수정한다.

```
local  all                                    crypt
host   all   127.0.0.1        255.255.255.255  crypt
host   all   210.110.144.169  255.255.255.255  crypt
```

참고로 host 로 시작하는 라인은 DB 서버에 접근하려는 클라이언트를 제어하려 한다. 그러 므로 DB 서버의 IP 를 주는 것이 아님을 주의하기 바란다. 또한 "**crypt**"는 DB 서버로의 접 근 제어 메소드이므로 "crypt"를 적용한 후부터는 DB서버로의 접속을 위해 계정과 패스워 드가 필요하다.

3. 다시 포스트그레스 슈퍼계정으로 "template1" 데이터베이스에 접속한 후 새로운 데이터베 이스 유저를 생성한다. 새로 생성할 유저를 "nogadax" 라고 가정하며 이 유저에게 데이터베 이스를 생성할 수 있는 권한을 줄 것이며 이 옵션은 "**createdb**" 이다. 기타 "**createuser**" 라 는 옵션도 있는데 이 옵션은 다른 유저를 생성할 수 있는 권한을 준다. 또한, "template1" 에 접속을 위한 옵션 "**-u**"를 주어야 하며 옵션을 준 후부터는 계정과 패스워드를 묻는다. 이 옵션을 주는 이유는 pg_hba.conf 에 "crypt" 접근 제어 메소드를 주었기 때문이다.

```
postgres]$ psql -u template1
```

```
Username : postgres
Password :
.........
template1=> create user nogadax with password 'qwert' createdb ;
CREATE  USER
template1=>\q
postgres]$
```

4. 생성된 nogadax 유저로 template1 데이터베이스에 접속한 뒤 ”nogadax”의 데이터베이스 를 생성하자. 데이터베이스명은 ”test” 라고 가정한다.

```
postgres]$ psql -u template1
Username : nogadax
Password :
..............
template1=>create database test;
CREATE DATABASE
template1=>\q
postgres]$ psql -u test
Username : nogadax
Password :
................
test=> \q
postgres]$
```