Real Alternative DBMS ALTIBASE, Since 1999

ALTIBASE & TOMCAT 연동 가이드

ALTIBASE 5

2010. 01

Document Control

Change Record

Date	Author	Change Reference
2010-01-25	snkim	Created
2014-03-31	Uijin.lee	modify

Reviews

Date	Name (Position)

Distribution

Name	Location

목차

개요

본 문서는 ALTIBASE와 TOMCAT 간 연동하는 방법을 기술한 문서로 TOMCAT은 5.5 와 6.0 버전, ALTIBASE는 5 버전을 대상으로 작성되었다.

TOMCAT 설치

TOMCAT 설치 방법에 대해 간단히 살펴보도록 한다. 환경은 Windows이고 TOMCAT 6.0 제품을 설치한다.

TOMCAT을 설치하기 전에 반드시 JRE(Java Runtime Environment) 혹은 JDK(Java Development Kit)가 설치되어 있어야한다.

TOMCAT 다운로드

http://tomcat.apache.org/download-60.cgi 사이트에 방문하여 최근 release된 TOMCAT 바이너리 파일을 다운받는다.

예를 들어, 위 홈페이지에서 apache-tomcat-6.0.24.zip 파일을 다운받는다.

TOMCAT 설치

다운로드 한 압축 파일을 적절한 디렉토리에 푼다.

환경변수 설정

다음의 환경변수를 설정한다.

1. CATALINA_HOME

   TOMCAT을 설치한 디렉토리를 지정한다.

2. JAVA_HOME 혹은 JRE_HOME

   JDK 혹은 JRE가 설치된 디렉토리를 지정한다.

3. PATH

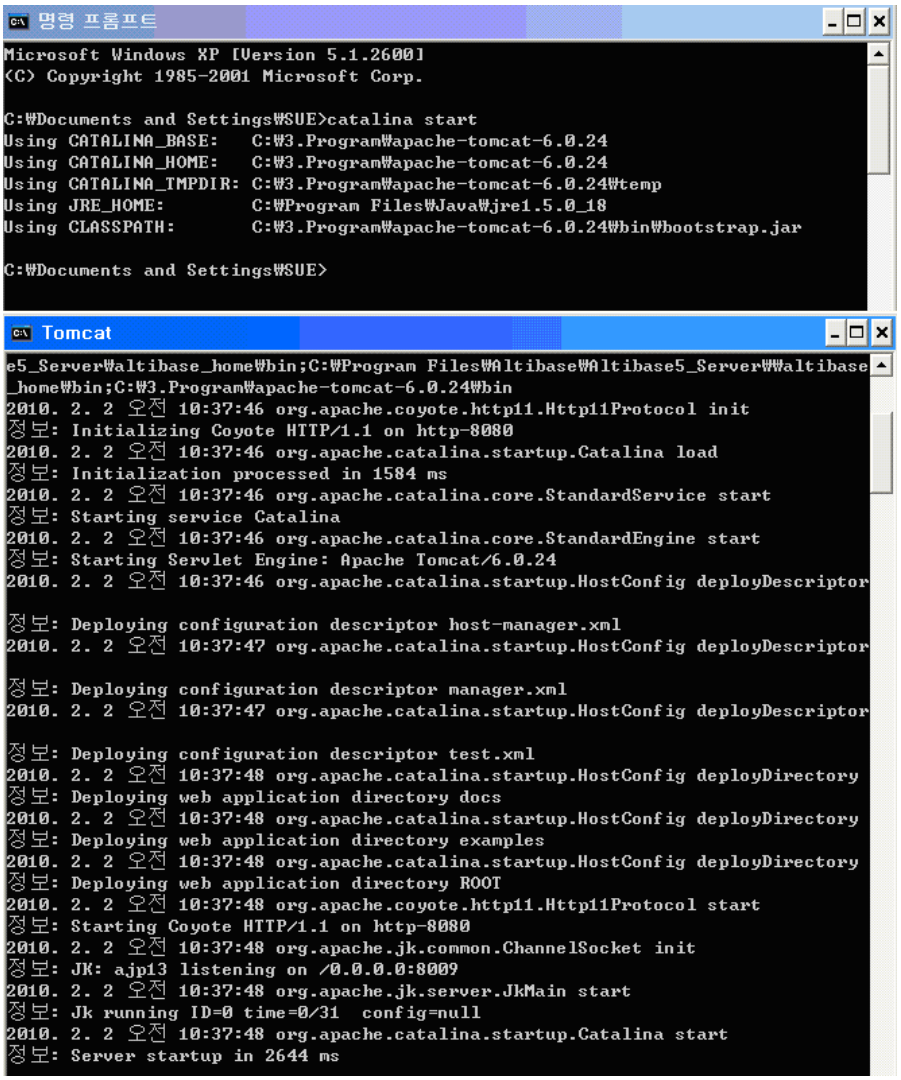
   $CATALINA_HOME/bin 디렉토리(1 번에 설정한 TOMCAT설치디렉토리 하위의 bin디렉토리의 경로)를 PATH에 추가한다.

TOMCAT 구동

다음의 명령어를 수행하여 TOMCAT을 구동시킨다.

$ catalina start

해당 명령을 실행하면 TOMCAT 구동내용이 display되는 별도의 창이 뜬다.

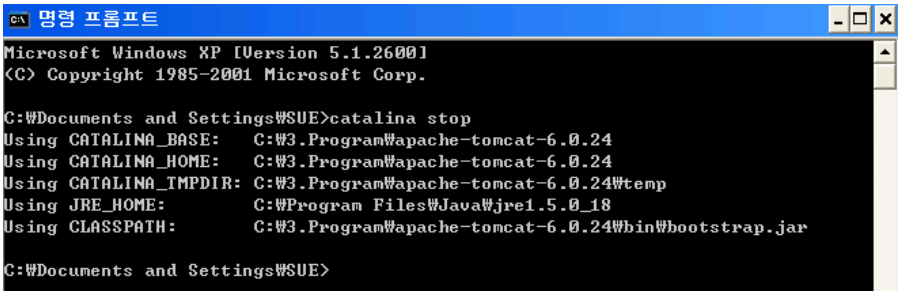


<그림 1> TOMCAT 구동 화면

TOMCAT 종료

다음의 명령어를 수행하여 TOMCAT을 종료시킨다.

$ catalina stop



<그림 2> TOMCAT 종료 화면

JDBC driver 설정

TOMCAT에서 ALTIBASE JDBC driver를 적절한 곳에 위치시켜 놓아야 ALTIBASE와 연동이 가능하다.

ALTIBASE JDBC driver 파일을 얻는 방법

ALTIBASE에서 제공하는 JDBC driver는 Altibase.jar 이다. 이 파일은 ALTIBASE가 설치되어있는 서버의 $ALTIBASE_HOME/lib 디렉토리 안에 존재한다.

Altibase 5 버전부터는 $ALTIBASE_HOME/lib 디렉토리에 Altibase.jar와 Altibase5.jar 파일이 존재하는데, Altibase.jar는 일반 JDBC driver 파일이며, Altibase5.jar는 ALTIBASE 5 버전과 그 이하의 버전을 함께 연동하고 싶을 때 사용한다. 따라서 하나의 ALTIBASE DB와 연동한다거나, 동일한 버전의 여러대의 ALTIBASE DB와 연동하기를 원한다면 일반적으로 $ALTIBASE_HOME/lib/Altibase.jar 파일을 사용하면 된다.

ALTIBASE JDBC driver 버전을 확인하는 방법

ALTIBASE JDBC driver 버전을 확인하는 방법은 다음의 명령어를 수행하면 된다.

```
$ java –jar Altibase.jar

JDBC Driver Info :    Altibase Ver = 5.3.1.7 for JavaVM v1.4, CMP:5.5.1, $Revision: 14502 $ Jan   3 2010 08:38:29
```

연동하려는 ALTIBASE DB Server와 ALTIBASE JDBC Driver가 호환 가능한지 확인을 위해 ALTIBASE DB Server 버전 확인이 필요한다. 이때, ALTIBASE DB Server의 cm protocol version과 ALTIBASE JDBC Driver의 CMP가 동일하면 호환 가능하다.

```
$ altibase -v
version 5.3.1.7 IBM_AIX_5.3-64bit-5.3.1.7-release-xlC_r (powerpc-ibm-aix5.3.0.0) Jan   3 2010 08:38:40, binary db version 5.3.1, meta version 5.5.1, cm protocol version 5.5.1, replication protocol version 5.3.1
```

버전이 UP 되면서 JDBC 관련 버그가 fix되었을 가능성이 있으므로, 일반적으로 ALTIBASE DB Server의 버전과 ALTIBASE JDBC driver버전을 동일하게 맞추거나 최신의 ALTIBASE JDBC driver 파일을 사용하는 것을 권장한다.

JDBC driver 파일을 TOMCAT에 위치

TOMCAT에서 ALTIBASE와 연동하기 위해 ALTIBASE JDBC driver(Altibase.jar)를 적절한 곳에 위치시켜 TOMCAT에서 자동으로 인식할 수 있도록 해야하는데 다음 중 한가지 방법으로 설정하면 된다.

1. Altibase.jar를 CLASSPATH에 추가
2. Altibase.jar를 $JAVA_HOME/jre/lib/ext 디렉토리 안에 위치
3. Altibase.jar를 $JRE_HOME/lib/ext 디렉토리 안에 위치
4. Altibase.jar를 $CATALINA_HOME/common/lib(Ver5.5), $CATALINA_HOME/lib(Ver6.0) 디렉토리 안에 위치

ALTIBASE와 TOMCAT의 연동

ALTIBASE와 TOMCAT을 연동하는 방법에 대해 기술한다. 연동하는 방법으로는 DBCP를 이용하여 연동, 일반 JDBC를 이용하여 연동, ATLIBASE의 ConnectionPool을 이용하여 연동하는 방법이 있다.

DBCP 방법으로 연동

DBCP(Database Connection Pool)는 TOMCAT에서 제공하는 Connection Pool을 말한다.

보통 JDBC를 이용하여 DB connection을 맺을 때 시간이 오래 걸리기 때문에, 이 비용을 줄이기 위해 일정수의 connection을 pool에 미리 맺어놓고 필요시마다 Connection을 가져와 사용하고 사용한 후 반납하는 구조이다. DBCP를 이용하면 데이터베이스에 접속할 때 마다 connection 객체를 생성할 필요가 없어진다.

JNDI Datasource를 구성할 때 <Resource> 태그에 정의하는 속성 정보에 대해 알아본다.

DBCP를 이용하여 ALTIBASE에 접속하기 위해서는 ALTIBASE용 DataSource를 JDNI에 미리 등록해놓고 웹 어플리케이션에서 이를 lookup하여 사용하면 된다.

연동 순서는 다음과 같다.

1. JNDI DataSource를 구성한다.
   - 1-1. 모든 웹 어플리케이션에서 공유하여 사용하도록 정의
     $CATALINA_HOME/conf/server.xml 파일의 < GlobalNamingResources> 자식노드로 <Resource> 태그를 정의한다.
   - 1-2. 설정한 웹 어플리케이션에서만 사용하도록 정의
     웹 어플리케이션 context.xml 파일의 < Context> 자식노드로 <Resource> 태그를 정의한다.
2. JNDI Resource 사용 설정
   - 2-1. 웹어플리케이션의 web.xml 파일을 구성한다.
     웹어플리케이션의 WEB-INF/web.xml 파일의 <web-app> 자식노드로 <resource>태그를 정의한다.
   - 2-2. context.xml 파일을 구성한다.
     JNDI DataSource를 GlobalNamingResources로 정의되어있다면, context.xml 파일의 < Context> 자식노드로 <ResourceLink> 태그를 정의한다.
3. 프로그램에서 DataSource를 사용하여 Connection을 얻어온다

각 단계에 대한 자세한 설명은 아래 사항을 참고한다.

1. JNDI DataSource 구성

JNDI DataSource를 구성하는 방법은 TOMCAT의 전체 웹 어플리케이션에서 사용할 수 있도록 정의하는 방법과 특정 웹 어플리케이션에서만 사용할 수 있도록 정의하는 방법이 있다.

먼저 전체 웹 어플리케이션에서 사용할 수 있도록 정의하는 방법으로는 $CATALINA_HOME/conf/server.xml에 JNDI DataSource를 구성하는 방법으로 JNDI DataSource에 관한 정보를 <GlobalNaminngResources> 자식노드로 <Resource> 태그를 정의하는 것이다. 이렇게 정의하면 특정 웹 어플리케이션에 국한 되어서만 사용되는 것이 아니라, TOMCAT의 전체 웹 어플리케이션에서 사용할 수 있다.

```
< GlobalNamingResources>
        <Resource name="jdbc/Altibase" auth="Container"
                type="javax.sql.DataSource"
                driverClassName="Altibase.jdbc.driver.AltibaseDriver"
                url="jdbc:Altibase://192.168.1.35:21129/mydb"
                username="sys" password="manager"
                maxActive="3" maxIdle="2" initialSize="1"
                defaultAutoCommit="false"
                removeAbandoned="true" logAbandoned="true"
                removeAbandonedTimeout="60"
                validationQuery="select 1 from dual"/>
</GlobalNamingResources>
* GlobalNamingResources/server.xml 파일 참조
```

또 다른 방법으로 설정한 웹 어플리케이션에서만 사용할 수 있도록 JNDI DataSource를 설정하는 것인데 특정 웹 어플리케이션의 META-INF/context.xml 파일의 < Context > 자식노드로 <Resource> 태그를 정의하면 해당 웹 어플리케이션에서만 사용이 가능하다.

```
<Context>
        <Resource name="jdbc/Altibase1" auth="Container"
                type="javax.sql.DataSource"
                driverClassName="Altibase.jdbc.driver.AltibaseDriver"
                url="jdbc:Altibase://192.168.1.35:21129/mydb"
                username="sys" password="manager"
                maxActive="3" maxIdle="2" initialSize="1"
                    defaultAutoCommit="false"
                    removeAbandoned="true"   logAbandoned="true"
                   removeAbandonedTimeout="60"
                    validationQuery="select 1 from dual"/>
</ Context >
* WebApplicationContext/META-INF/META-INF/context.xml 파일 참조
```

<Resouce> 태그에 정의하는 DBCP의 속성은 다음과 같다.

driverClassName	Altibase JDBC driver class Name
url	ALTIBASE와 연결을 위한 Connection string정보 jdbc:Altibase://IP:port_no/db_name" 형태로 기입
username	데이터베이스 계정
password	데이터베이스 패스워드
maxActive	최대 Connection 수, 0 은 무제한. default는 8
initialSize	초기 Connection 수. default는 0
maxIdle	Pool에 idle하게 유지하는 최대 연결 수. default는 8
minIdle	Pool에 idle하게 유지하는 최소 연결 수. default는 0
maxWait	최대 연결 시도 시간 (단위 : millisec) -1 은 무한 대기 Default는 무한 대기
validationQuery	연결의 validation을 체크하기 위해 사용하는 SQL문 반드시 최소한 1 개이상의 row가 return되는 select문으로 지정 Ex) select 1 from dual
defaultAutoCommit	Autocommit 모드를 설정. default는 true
defaultTransactionIsolation	Transaction Isolation level을 설정한다. NONE, REPEATABLE_READ, SERIALIZABLE 의 값을 설정할 수 있고, default는 DB서버의 default 값을 따른다. ALTIBASE의 isolation level은 default 로 READ COMMITTED 이다.
removeAbandoned	버려져서 사용되지않는 Connection에 대한 제거기능을 결정하는 속성. default 는 false Connection이 제거되는 시점은 Connection 개수가 maxActive설정된 값을 초과하여 Connection을 요청한 이후이다. maxActive에 도달하지 않은 시점에는 Connection을 할당받을 수 있으므로, 사용되지않는 Connection 제거가 일어나지 않는다.
removeAbandonedTimeout	버려져서 사용되지않는 connection에 대한 제거가 일어나는 시간을 설정하는 속성. default로 300 초
logAbandoned	버려져서 사용되지않는 connection에 대한 제거시 로그에 stack정보를 남길지를 결정하는 속성. default로 false

2. JNDI Resource 사용 설정

일반적으로 JNDI Resource를 사용하기위해 웹어플리케이션의 WEB-INF/web.xml 파일의 <web-app> 자식노드로 <resource-ref>태그를 정의한다.

```
<resource-ref>
        <description>Altibase Test</description>
        <res-ref-name>jdbc/Altibase1</res-ref-name>
        <res-type>javax.sql.DataSource</res-type>
        <res-auth>Container</res-auth>
  </resource-ref>
```

* res-ref-name의 값은 위의 JNDI DataSource 설정 단계에서 정의한 <Resource> 태그의 name 속성 값과 동일해야 한다.

* WebApplicationContext/WEB-INF/web.xml파일 참조

만약, JNDI DataSource가 GlobalNamingResources로 정의되어있다면 $CATALINA_HOME/conf/context.xml 파일 혹은 웹 어플리케이션 META-INF/context.xml 파일의 < Context> 자식노드로 <ResourceLink> 태그를 정의해서 JNDI DataSource를 사용할 수 있다. CATALINA_HOME/conf/context.xml 파일에 정의할 경우에는 TOMCAT 전체의 웹 어플리케이션에서 JNDI Datasource를 사용할 수 있고, 웹 어플리케이션의 context.xml 파일에 정의할 경우에는 설정한 웹 어플리케이션에서만 JNDI Datasource를 사용할 수 있다.

```
<Context>
    <ResourceLink global="jdbc/Altibase" name="jdbc/Altibase"
                  type="javax.sql.DataSource"/>
</Context>
```

* global과 name의 값은 위의 JNDI DataSource 설정 단계에서 정의한 <Resource> 태그의 name 속성 값과 동일해야 한다.

* GlobalNamingResources/ GlobalNamingResources /META-INF/context.xml 파일 참조

3. 프로그램에서 Connection 얻는 방법

프로그램에서 다음과 같이 JNDI Datasource를 통해 Connection을 얻는다.

이때, java:/comp/env는 JNDI를 lookup할 때 사용하는 prefix이다.

```
Context envContext  = (Context)new InitialContext().lookup("java:/comp/env");
DataSource ds = (DataSource)envContext.lookup("jdbc/Altibase");
Connection conn = ds.getConnection();
```

또는

```
Context initCtx = new InitialContext();
```

```
DataSource ds =    (DataSource)initCtx.lookup("java:comp/env/jdbc/Altibase");
Connection conn = ds.getConnection();
```

다음은 위에 설정한 JNDI Datasource를 이용하여 ALTIBASE와 연동하는 jsp 예제 프로그램이다.

```
<%@ page import="java.sql.*, javax.naming.*, javax.sql.*"%>
<%
   Context initCtx = new InitialContext();
   DataSource ds =    (DataSource)initCtx.lookup("java:comp/env/jdbc/Altibase");
   Connection conn = ds.getConnection();
   Statement stmt = conn.createStatement();
   String query = "select to_char(sysdate,'yyyy/mm/dd hh24:mi:ss') from dual";
   ResultSet rs = stmt.executeQuery(query);
   if(rs.next()) {
       out.println(rs.getString(1));
   }
   rs.close();
   stmt.close();
   conn.close();0
%>
* GlobalNamingResources/GlobalNamingResources /test.jsp 파일 참조
```

일반 JDBC 방법으로 연동

1. JDBC driver 로딩한다.

   Class.forName("Altibase.jdbc.driver.AltibaseDriver");

2. Connection 객체 얻는다.

   DriverManager 를 통해 Connection을 얻어오는데 연결 URL 정보는 " jdbc:Altibase://IP:port_no/db_name "이다.

   java.util.Properties props = new java.util.Properties();

   props.put("user","sys"); //데이터베이스 계정

   props.put("password","manager"); //데이터베이스 패스워드

   String url = " jdbc:Altibase://127.0.0.1:20300/mydb";

   Connection conn = DriverManager.getConnection(url,props);

3. Statement를 객체를 얻는다.

   Statement stmt = conn.createStatement();

4. SQL문을 실행한 후 ResultSet에 담는다.

   String sql = “select to_char(sysdate,’yyyy/mm/dd hh:mi:ss’) from dual”;

   ResultSet rs = stmt.executeQuery(sql);

5. 작업이 완료된 후 관련된 모든 객체를 close한다.

   rs.close();

   stmt.close();

   conn.close();

다음은 일반 JDBC를 이용하여 ALTIBASE와 연동하는 jsp 예제 프로그램이다.

```
<%@ page import="java.sql.* "%>
<%
    Class.forName("Altibase.jdbc.driver.AltibaseDriver");
    java.util.Properties props = new java.util.Properties();
    props.put("user","sys");
    props.put("password","manager");
    String url = "jdbc:Altibase://127.0.0.1:20300/mydb";
    Connection conn = DriverManager.getConnection(url,props);
    Statement stmt = conn.createStatement();
    String sql = "select to_char(sysdate,'yyyy/mm/dd hh:mi:ss') from dual";
    ResultSet rs = stmt.executeQuery(sql);
    if(rs.next()) {
        out.println(rs.getString(1));
    }
    rs.close();
    stmt.close();
    conn.close();
%>
```

ALTIBASE의 ConnectionPool을 이용하여 연동

ALTIBASE의 ConnectionPool을 이용하는 방법은 ABConnectionPoolDataSource 객체를 이용하여 Connection을 얻어오면 된다. ABConnectionPoolDataSource 클래스는 Altibase.jar 파일에 포함되어 있으며, Altibase.jdbc.driver 패키지 안에 존재한다.

ConnectionPool을 jsp를 로딩할 때, 혹은 특정 servlet을 호출할때마다 생성하는 것이 아니라, 웹어플리케이션에서 공유할 수 있도록 ServeltConext에 등록하여 사용하는 것을 권장한다.

ABConnectionPoolDataSource을 이용하여 ConnectionPool을 생성하고 이를 ServeltConext에 등록하고 jsp 페이지에서 사용하는 방법은 다음과 같다.

1. ServletContextListener 구현

   ServletContextListener 인터페이스를 implements하는 클래스를 구현한다.

   해당 클래스의 위치는 웹 어플리케이션/WEB-INF/classes 폴더에 위치시킨다.

   contextInitialized 메소드안에서 ABConnectionPoolDataSource를 생성한후 ABConnectionPoolDataSource클래스의 setUrl 메소드를 호출하여 Connection url을 셋팅한다.

```
import javax.servlet.*;
public class ConnectionPool implements ServletContextListener {
   private Altibase.jdbc.driver.ABConnectionPoolDataSource mPool;
        public void contextInitialized(ServletContextEvent sce){
                ServletContext context = sce.getServletContext();
                String driverName = context.getInitParameter("driverName");
                String url = context.getInitParameter("url");
                String user = context.getInitParameter("user");
                String password = context.getInitParameter("password");

                mPool= new Altibase.jdbc.driver.ABConnectionPoolDataSource();
                try{
                    Class.forName(driverName);
                    mPool.setInitialPoolSize(5);
                    mPool.setUrl(url+"?user="+user+"&password="+password);
                    context.setAttribute("connectionPool",mPool);
                }catch(Exception e){
                   System.out.println(e);
                }
        }
        public void contextDestroyed(ServletContextEvent sce){
                 if(mPool != null){
                       mPool.close();
                 }
        }
```

```
}
```

* contextInitialized 메소드는 웹어플리케이션이 로딩될 때 웹 컨테이너에 의해 자동으로 호출되는 메소드이다.

* ServletContext의 setAttribute 메소드를 이용하여 ABConnectionPoolDataSource 객체를 등록하면, 웹 어플리케이션에서 공유하여 사용할 수 있다.

* AltibaseConnectionPool/WEB-INF/classes/ConnectionPool.java 파일 참조

2. web.xml에 servletContextListener 등록

ServletContextListener 구현한 클래스를 웹 어플리케이션이 로딩될 때 자동으로 호출하기 위해서는 web.xml 파일에 <listener> 태그를 이용하여 등록해야한다.

이때 <context-param> 태그를 이용하면 ServeltConext의 getInitParameter 메소드를 호출하여 param 값을 가지고 올 수 있다.

```
<context-param>
    <param-name>driverName</param-name>
    <param-value>Altibase.jdbc.driver.AltibaseDriver</param-value>
  </context-param>
  <context-param>
    <param-name>url</param-name>
    <param-value>jdbc:Altibase://127.0.0.1:20300/mydb</param-value>
  </context-param>
  <context-param>
    <param-name>user</param-name>
    <param-value>sys</param-value>
  </context-param>
  <context-param>
    <param-name>password</param-name>
    <param-value>manager</param-value>
  </context-param>

  <listener>
    <listener-class>ConnectionPool</listener-class>
 </listener>
```

* <context-param> : ServletContext에서 사용할 parameter값을 지정

* <listener-class> : ServletContextListener 구현한 클래스

* AltibaseConnectionPool/WEB-INF/web.xml 파일 참조

3. jsp에서 connection사용

ServletConext객체는 jsp의 내장객체중 application 객체로 표현된다.

application으로부터 ABConnectionPoolDataSource 객체를 얻어와 ABConnectionPoolDataSource 객체의 getConnection 메소드를 호출해 Connection 객체를 얻어올 수 있다.

```
<%@ page import="java.sql.*,javax.sql.* "%>
<%
    DataSource mPool = (DataSource)application.getAttribute("connectionPool");
    Connection conn = mPool.getConnection();
    Statement stmt = conn.createStatement();
    String sql = "select to_char(sysdate,'yyyy/mm/dd hh:mi:ss') from dual";
    ResultSet rs = stmt.executeQuery(sql);
    if(rs.next()) {
        out.println(rs.getString(1));
    }
    rs.close();
    stmt.close();
    conn.close();
%>
```

* AltibaseConnectionPool/test.jsp 파일 참조

FailOver Configuration

ALTIBASE가 제공하는 FailOver 기능을 이용하여 TOMCAT과 연동하는 방법에 대해 기술한다.

FailOver Configuration

TOMCAT은 FailOver 기능을 제공하지 않는다. 하지만 ALTIBASE 5.3.3 버전부터 FailOver 기능을 제공하기 때문에, ALTIBASE의 FailOver기능을 이용하면 FailOver가 구현 가능하다.

사용방법은 다음과 같다.

1. server.xml 파일에 JNDI DataSource를 지정할 때 Connection url부분에 FailOver관련 속성을 지정해준다.

```
< GlobalNamingResources>
    <Resource name="jdbc/Altibase" auth="Container" type="javax.sql.DataSource"
              driverClassName="Altibase.jdbc.driver.AltibaseDriver"
              maxActive="100" maxIdle="-1" maxWait="-1"
              username="sys" password="manager"
              url="jdbc:Altibase://192.168.6.224:21129/mydb?
                   AlternateServers=(192.168.1.35:21129)&amp;
                   ConnectionRetryCount=3&amp;
                   ConnectionRetryDelay=3&amp;
                   SessionFailOver=on&amp;
                   LoadBalance=off"
              validationQuery="SELECT 1 FROM DUAL" testOnBorrow="true"
              removeAbandoned="true" removeAbandonedTimeout="60"
              logAbandoned="true" />
</GlobalNamingResources>
* FailOver/server.xml 파일 참조
```

FailOver 관련 속성은 다음과 같다.

AlternateServer	장애 발생시 접속하게 될 가용 서버를 나타내며 (IP Address1:Port1, IP Address2:Port2,...) 형식으로 기술한다.
ConnectionRetryCount	가용 서버 접속 실패 시, 접속 시도 반복 횟수
ConnectionRetryDelay	가용 서버 접속 실패 시, 다시 접속을 시도하기 전에 대기하는 시간(초단위)
LoadBalance	on으로 설정하면 최초 접속 시도 시에 기본 서버와 가용

	서버를 포함하여 랜덤으로 선택한다. off로 설정하면 최초 접속 시도 시에 기본 서버에 접속하고, 접속에 실패하면 AlternateServer로 기술한 서버에 접속한다.
SessionFailOver	STF(Service Time Fail-Over)를 할 것인지 여부를 나타낸다. on : STF, off : CTF CTF(Connection Time Fail-Over)는 DBMS 접속 시점에 장애를 인식하여 장애가 발생한 DBMS대신 다른 가용 노드의 DBMS로 접속하고 서비스를 진행한다. STF(Service Time Fail-Over)는 DBMS 접속에 성공하여 서비스하는 도중에 장애가 발생하는 것으로, 다른 가용 노드의 DBMS에 다시 접속하여 세션의 프로퍼티를 복구한 후 사용자 응용 프로그램의 업무 로직을 다시 수행하도록 하는 것을 의미한다. 즉 장애가 발생한 DBMS에서 수행된 작업을 다시 한 번 수행할 필요가 있는 경우이다.

CTF 및 STF 의 구현방법은 ALTIBASE FailOver 기술문서를 참조한다.

TOMCAT 연동시 주의사항

TOMCAT 연동시 주의사항에 대해 설명한다.

Tomcat 사용시 JDK 환경

DBCP를 이용하여 preparestatement를 처리할 때 고려해야 할 사항이 있는데 성능 향상을 위해 Tomcat 설정 중 testOnBorrow (default : true), poolPrepareStatements (default : false) 설정을 변경 사용할 경우 되도록 JDK(JRE) 1.6 이상의 환경에서 사용하길 권장한다.

문제의 증상

Tomcat log (version 6)

```
--- Check the statement (query failed).

--- Cause: java.sql.SQLException: [0]:Failure to find statement; nested exception is
com.ibatis.common.jdbc.exception.NestedSQLException:

--- The error occurred in maps/CommonSqlMap.xml.

--- The error occurred while applying a parameter map.

--- Check the Common.getIservSp-InlineParameterMap.

--- Check the statement (query failed).

--- Cause: java.sql.SQLException: [0]:Failure to find statement
```

정상적인 경우

Tomcat log (version 6)

```
[0001][14:14:31 386][    2][    0] GET-
CONNECTION[org.apache.tomcat.dbcp.dbcp.BasicDataSource.getConnection] [1 ms]
```

확인 방법

$CATALINA_HOME/conf/server.xml 파일 내용 중 아래 부분의 주석을 제거하신 후 Tomcat을 재 시작하여 $CATALINA_HOME/logs/tomcat.log 를 확인합니다.

변경전>

```
<!--
<Valve className="org.apache.catalina.valves.AccessLogValve"
       directory="logs"   prefix="localhost_access_log." suffix=".txt"
       pattern="common" resolveHosts="false"/>
-->
```

변경후>

```
<Valve className="org.apache.catalina.valves.AccessLogValve"
       directory="logs"   prefix="localhost_access_log." suffix=".txt"
       pattern="%a %b %t %s %D %r " resolveHosts="false"/>
```

참고 : http://tomcat.apache.org/tomcat-5.5-doc/config/valve.html

권장 Tomcat 설정 값

	환경 1	환경 2	환경 3
JDK Ver	1.5	1.6	1.6
Tomcat Ver	6 (dbcp ver 1.3)	6 (dbcp ver 1.3)	7 (dbcp ver 1.4)
testOnBorrow	TRUE 만 권장 (Defalut)	True/False	True/False
poolPrepareStatements	FALSE 만 권장 (Defalut)	True/False	True/False
Result	testOnBorrow=False, poolpreparestatement=true 사용시 권장하지 않음, JRE 또는 JDK1.6 으로 업그레이드 권장	권장	권장

* **testOnBorrow**

  + connection pool에서 connection을 가져올 때 해당 connection이 유효성 검사 여부
  + 기본값은 false이며, 일반적으로 기본값을 사용한다. true설정하게 되면 매번 validationQuery를 수행하기 때문에 약간의 성능저하를 감수해야 한다.

* **poolpreparestatement**

  + DBCP에서 Statement 풀링이 커넥션 별로 유지되고 있기 때문에, 응용서버(App)에서는 질의를 컴파일 한 정보를 계속 유지하게 된다. 이 후 같은 질의에 대해서는 prepare 과정 없이 execute 단계만 반복함으로써 성능 향상을 가져올 수 있다.

사용한 Resource의 반납

프로그램에서 할당한 Connection, Statement, ResultSet는 사용이 끝나면 반드시 명시적으로 close해줘야한다. 만약 close해주지 않으면 위의 Resource를 선언한 변수의 life cycle동안 계속 Resource를 할당하고 있기 때문에 불필요한 Resource 소모를 가져오게 된다.

Connection을 close해주지 않으면 pool로 Connection이 반납되지 않기 때문에 다른 곳에서 Connection을 사용하지 못할 것이다.

Statement는 session이 유지되는 동안 어플리케이션 뿐만 아니라, DB 서버상에도 Statement 정보가 메모리에 할당되어 남아있게 된다. 따라서 Statement를 close해주지 않고 Connection도 close해주지 않는다면, 어플리케이션에서 불필요한 Resource가 계속 할당되는 것도 문제지만, DB 서버의 Query_Prepare 메모리 영역도 증가하게 된다.

만약 위의 Resource들을 명시적으로 close해주지 않았다면, 해당 Resource를 선언한 변수의 life cycle동안 메모리에 할당되어 있다가 life cycle이 끝나면 해당 Resource를 더 이상 참조하지 않게 되므로, JVM(Java Virtual Machine)의 GC(Garbage Collector)가 후에 해당 Resouce들을 해제하게 될 것이다. 하지만 일반적으로 GC는 우선순위가 가장 낮은 쓰레드이다. 따라서 GC가 Resource를 해제하는 시점이 언제인지는 예측하기가 어렵다. 즉, 어플리케이션에서 변수의 life cycle이 끝나서 더 이상 참조되지 않기 때문에 바로 해제될 수 있는 것이 아니고, GC가 해제하지 않는 이상 DB 서버에서는 계속 해당 Resource에 대한 정보를 가지고 있게 된다.

따라서, 반드시 사용이 끝난 Resource는 명시적으로 close 해주는 것을 권장한다.

버려진 Connection을 제거

DBCP를 이용하여 connection을 처리할때 문제점이 하나 있는데 바로 웹어플리케이션에서 명시적으로 ResultSet, Statement, Connection을 close해줘야 한다는 것이다. 만약 웹어플리케이션에서 이러한 resource를 close하는 것을 실패했다면 이후에 resource들을 재사용할 수 없게 된다. 이러한 현상을 Connection pool leak이라고 부르며, connection pool leak이 지속된다면 최종에는 이용가능한 connection이 모두 없어지게 될 것이다. 이러한 문제를 막기위해 DBCP에서 문제가 발생하여 버려진 connection을 복구하고 로깅하는 방법을 제공하는데 다음의 속성을 설정하면 된다.

```
removeAbandoned="true"
```

TOMCAT 연동시 오류사항

TOMCAT에서 ALTIBASE 연동시 자주 발생되는 오류사항에 대한 내용을 설명한다.

No suitable driver

java.sql.SQLException: No suitable driver 오류는 접속 URL정보중 Altibase에 해당하는 부분을 잘못 기입했을 때 발생한다.

```
url="jdbc:mysql://127.0.0.1:20300/mydb"
```

jdbc:Altibase 부분이 jdbc:mysql 부분으로 잘못 설정되어있기 때문에 에러가 발생한다.

Communication link failure

java.sql.SQLException: Communication link failure 오류는 다음과 같은 상황에서 발생한다.

1. ALTIBASE DB server가 구동중이지 않는 상황
2. ALTIBASE JDBC driver 파일이 ALTIBASE DB server의 버전과 상이한 경우

   예) ALTIBASE DB Server의 버전은 5.3.1 버전인데, 5.1.1 버전의 ALTIBASE JDBC driver 파일을 사용하는 경우
3. 접속 URL 정보가 잘못 설정된 경우

   3-1. IP가 잘못 설정된 경우

   3-2. PORT_NO가 잘못 설정된 경우

   예) ALTIBASE DB Server의 IP주소가 10.10.10.10 이고 PORT_NO가 20300 인데 IP주소를 10.10.10.1 로 설정하고나 PORT_NO를 20301 로 설정한 경우

```
url="jdbc:Altibase://10.10.10.1:20301/mydb"
```

그외 확인 사항

간혹 웹 어플리케이션의 context.xml 파일에 DataSource를 지정했을 때 Connection url부분이 정확함에도 불구하고 Client unable to establish connection 에러 혹은, driver 클래스가 null이라는 에러를 접할 수도 있다. 이때는 $CATALINA_HOME/conf/catalina 디렉토리안의 웹어플리케이션이름.xml 파일을 지우도록 한다. 간혹 웹 어플리케이션의 context.xml 파일이 변경되었는데도 해당 파일이 reloading이 안되어 에러가 발생하는 경우가 있다.

첨부 예제 파일

첨부된 예제 파일에 대해 설명한다.

첨부 예제

1. AltibaseConnectionPool

   ABConnectionPoolDataSource을 이용하여 ALTIBASE가 제공하는 ConnectionPool을 이용하는 예제이다.

2. FailOver

   ALTIBASE 5.3.3 버전의 FailOver를 이용하는 예제이다.

3. GlobalNamingResources

   GlobalNaming JNDI DataSource를 이용하는 예제이다.

4. WebApplicationContext

   특정 웹 어플리케이션에서 DataSource를 이용하는 예제이다.

알티베이스㈜
서울특별시 구로구 구로 3 동 182-13
대륭포스트 2 차 1008 호
02-2082-1000
http://www.altibase.com

대전사무소
대전광역시 서구 둔산동 921
주은리더스텔 901 호
042-489-0330

기술지원본부
서울특별시 구로구 구로 3 동 182-13
대륭포스트 2 차 908 호
02-2082-1000

기술지원센터
02-2082-1114
http://support.altibase.com