

JSTL の SQL アクションを使った簡単なデータベース・クエリー

JSTL の SQL アクションを使って Apache Derby のデータにアクセスする

Meenakshi Guruaribam Khanna

2006年 8月 10日

プロトタイプ of データベースを構築をするにせよ、データベース・アクセスをテストするにせよ、データベース・クエリーは日々の Web 開発作業の一部です。データベースのクエリーや更新といった単純な操作は複雑であるべきではなく、時間のかかるものにするべきでもありません。いくつかの操作を 1 つのトランザクションにグループ化することについても同様です。この記事では、著者の Meenakshi G. Khanna が、単純な JSTL 1.1 SQL タグと Tomcat 5.5、そして Apache Derby データベースを使って、データベース・クエリーを容易に行う方法について解説します。

プロトタイプ of データベースを構築をするにせよ、データベース・アクセスをテストするにせよ、データベースへのクエリーには毎日関わらざるを得ません。データベースのクエリーや更新、あるいは複数の操作を 1 つのトランザクションにグループ化するという操作は、本来は単純であるべきものです。この記事では、JSTL (Java™ Server Pages Standard Tag Library) 1.1 の SQL タグ・ライブラリーを使って、Apache Derby リレーショナル・データベースの中にあるデータにアクセスし、操作する方法について解説します。この記事を読み進むためには、JDK 1.4 の他に、この記事の中で使われている下記の技術をダウンロードし、インストールする必要があります。

- JSTL ライブラリー: ライブラリーの内容をテンポラリーの場所 (C:\temp など) にダウンロードし、解凍します。
- Tomcat 5.5: JSTL 1.1 のメンテナンス・リリースには、Servlet 2.4 仕様と JSP 2.0 仕様をサポートする JSP コンテナが必要です。Apache Tomcat 5.x はこれらの仕様をサポートするため、この記事の例では Apache Tomcat 5.x を JSP コンテナとして使っています。
- Apache Derby 10.1.2.1: Apache Derby は、オープンソースのリレーショナル・データベース管理システムとして一般的なものです。この bin ディストリビューション・パッケージ、db-derby-10.1.2.1-bin.zip をダウンロードし、解凍します。\\docs\pdf\getstart フォルダの下にある PDF には、Apache Derby をインストールするための手順が説明されています。

この記事は、皆さんが JSP 技術や JSTL の SQL タグ、単純なデータベース・クエリーなどに慣れているものと想定しています。

J2SE 1.4 用の Tomcat

Tomcat 5.5 は J2SE 5.0 以上で実行するように設計されています。もし皆さんが J2SE 1.4 を実行している場合には、Tomcat のインストール・パッケージと一緒に、JDK1.4 互換性パッケージ (apache-tomcat-5.5.17-compat.zip) をダウンロードしてください(参考文献を参照)。互換性パッケージをインストールすると、J2SE 1.4 で Tomcat を実行するために必要な、jmx.jar と xercesImpl.jar、そして xml-apis.jar が追加されます。

Tomcat の中で JSTL をコンフィギュレーションする

まず、Tomcat の中で JSTL ライブラリーをコンフィギュレーションする必要があります。最初に、\$CATALINA_HOME\webapps\ROOT\WEB-INF\の下に、「tlds」というフォルダーを作成し、tlds ファイル、C:\temp\jakarta-taglibs\standard\tld\c.tld と C:\temp\jakarta-taglibs\standard\tld\sql.tld を、\$CATALINA_HOME\webapps\ROOT\WEB-INF\tlds ディレクトリーにコピーします。

次に、解凍された JAR ファイル、C:\temp\standard\lib\jstl.jar と C:\temp\standard\lib\standard.jar を、\$CATALINA_HOME\webapps\ROOT\WEB-INF\lib ディレクトリーにコピーします。CATALINA_HOME が Tomcat インストール・ディレクトリーを参照することに注意してください。

最後に、リスト 1 に示すエントリーを、\$CATALINA_HOME\webapps\ROOT\WEB-INF\web.xml デプロイメント記述子 (あるいは皆さんの Web アプリケーションの web.xml デプロイメント記述子) に追加します。

リスト 1. Tomcat のデプロイメント記述子のエントリー

```
<taglib>
  <taglib-uri> http://java.sun.com/jsp/jstl/core </taglib-uri>
  <taglib-location> /WEB-INF/tlds/c.tld </taglib-location>
</taglib>
<taglib>
  <taglib-uri> http://java.sun.com/jsp/jstl/sql </taglib-uri>
  <taglib-location> /WEB-INF/tlds/sql.tld </taglib-location>
</taglib>
```

Apache Derby をコンフィギュレーションする

Apache Derby には、ij (interactive java) というツールが付属しています。このツールを使うと、データベースを作成でき、また SQL ステートメントを実行することができます。ij ツールの実行は、次の手順で行います (Derby の zip パッケージを解凍したディレクトリーが <DERBY_INSTALL_HOME> であることに注意してください)。

1. PATH 変数に <DERBY_INSTALL_HOME>\frameworks\embedded\bin と JDK パス (例えば C:\jdk1.4\bin など) を含めます。
2. DERBY_INSTALL 変数を <DERBY_INSTALL_HOME> に設定します (例えば C:\Derby\db-derby-10.1.2.1-bin など)。
3. クラスパスを <DERBY_INSTALL_HOME>\lib\derby.jar と <DERBY_INSTALL_HOME>\lib\derbytools.jar に設定します。

また、derby.system.home という Java システム・プロパティを規定すると、Derby が起動する際にシステム・ディレクトリー (つまり、どんなデータベースが保存されているか、どこに新しいデータベースを作成するか、どんなコンフィギュレーション・パラメーターを使用するか、など) を定義することもできます。例えばこのディレクトリーを、derby.system.home=C:\Derby\databases のように設定することができます。Derby を起動する際にシステム・ディレクトリーを規定しないと、カレント・ディレクトリーがシステム・ディレクトリーになります。

データベースとスキーマを作成する

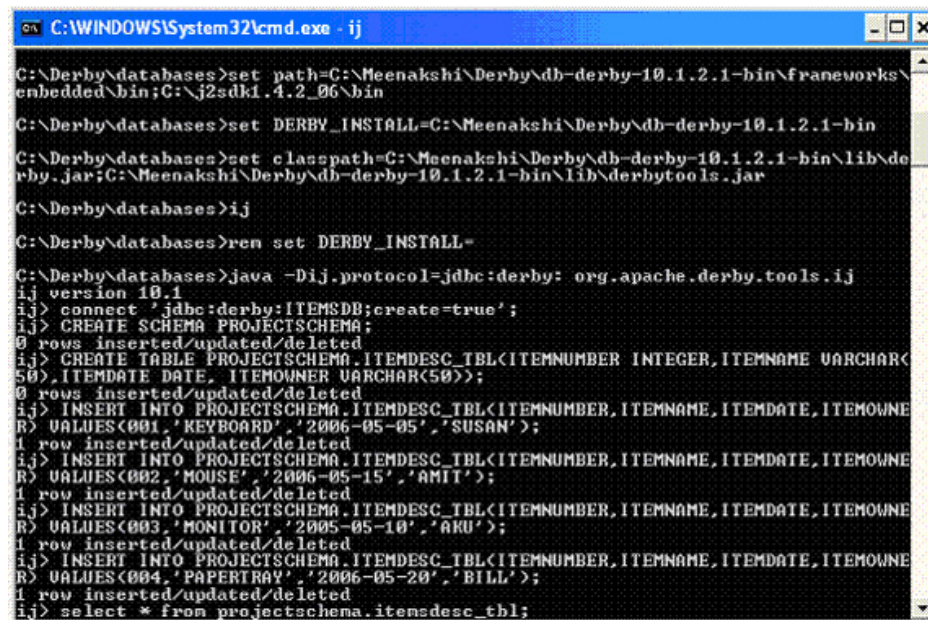
リスト 2 は、ITEMSDB というデータベースと、PROJECTSCHEMA というスキーマ、そして ITEMDESC_TBL というテーブルを、ij ユーティリティを使って作成する方法を示しています。

リスト 2. ij プロンプトで発行するコマンド

```
ij> connect 'jdbc:derby:ITEMSDB;create=true';
ij>CREATE SCHEMA PROJECTSCHEMA;
ij> CREATE TABLE PROJECTSCHEMA.ITEMDESC_TBL (ITEMNUMBER INTEGER,ITEMNAME VARCHAR(50),
ITEMDATE DATE, ITEMOWNER VARCHAR(50));
ij>INSERT INTO PROJECTSCHEMA.ITEMDESC_TBL (ITEMNUMBER,ITEMNAME,ITEMDATE,ITEMOWNER)
VALUES(001,'KEYBOARD','2006-05-05','SUSAN');
ij>INSERT INTO PROJECTSCHEMA.ITEMDESC_TBL (ITEMNUMBER,ITEMNAME,ITEMDATE,ITEMOWNER)
VALUES(002,'MOUSE','2006-05-15','AMIT');
ij>INSERT INTO PROJECTSCHEMA.ITEMDESC_TBL (ITEMNUMBER,ITEMNAME,ITEMDATE,ITEMOWNER)
VALUES(003,'MONITOR','2005-05-10','AKU');
ij>INSERT INTO PROJECTSCHEMA.ITEMDESC_TBL (ITEMNUMBER,ITEMNAME,ITEMDATE,ITEMOWNER)
VALUES(004,'PAPERTRAY','2006-05-20','BILL');
```

図 1 は、ij プロンプトからコマンドを発行する様子のスナップショットを示しています。

図 1. ij を使ってパスを設定し、SQL コマンドを発行する



```
C:\Derby\databases>set path=C:\Meenakshi\Derby\db-derby-10.1.2.1-bin\frameworks\
embedded\bin;C:\j2sdk1.4.2_06\bin
C:\Derby\databases>set DERBY_INSTALL=C:\Meenakshi\Derby\db-derby-10.1.2.1-bin
C:\Derby\databases>set classpath=C:\Meenakshi\Derby\db-derby-10.1.2.1-bin\lib\de
rby.jar;C:\Meenakshi\Derby\db-derby-10.1.2.1-bin\lib\derbytools.jar
C:\Derby\databases>ij
C:\Derby\databases>rem set DERBY_INSTALL=
C:\Derby\databases>java -Dij.protocol=jdbc:derby: org.apache.derby.tools.ij
ij version 10.1
ij> connect 'jdbc:derby:ITEMSDB;create=true';
ij> CREATE SCHEMA PROJECTSCHEMA;
0 rows inserted/updated/deleted
ij> CREATE TABLE PROJECTSCHEMA.ITEMDESC_TBL (ITEMNUMBER INTEGER,ITEMNAME VARCHAR(
50),ITEMDATE DATE, ITEMOWNER VARCHAR(50));
0 rows inserted/updated/deleted
ij> INSERT INTO PROJECTSCHEMA.ITEMDESC_TBL (ITEMNUMBER,ITEMNAME,ITEMDATE,ITEMOWNE
R) VALUES(001,'KEYBOARD','2006-05-05','SUSAN');
1 row inserted/updated/deleted
ij> INSERT INTO PROJECTSCHEMA.ITEMDESC_TBL (ITEMNUMBER,ITEMNAME,ITEMDATE,ITEMOWNE
R) VALUES(002,'MOUSE','2006-05-15','AMIT');
1 row inserted/updated/deleted
ij> INSERT INTO PROJECTSCHEMA.ITEMDESC_TBL (ITEMNUMBER,ITEMNAME,ITEMDATE,ITEMOWNE
R) VALUES(003,'MONITOR','2005-05-10','AKU');
1 row inserted/updated/deleted
ij> INSERT INTO PROJECTSCHEMA.ITEMDESC_TBL (ITEMNUMBER,ITEMNAME,ITEMDATE,ITEMOWNE
R) VALUES(004,'PAPERTRAY','2006-05-20','BILL');
1 row inserted/updated/deleted
ij> select * from projectschema.itemsdesc_tbl;
```

終了することを忘れないように

テーブルとレコードの作成が終了したら、必ず `ij>exit;` をタイプして、ij を終了します。2 つの別々の Derby のインスタンスが同じデータベースをアクセスしてはいけません。例えば埋め込

み環境では、Derbyデータベースにアクセスするアプリケーションがローカルの JDBC ドライバーを起動し、このドライバーがDerby のインスタンスを起動します。ij などの別のアプリケーションを起動して同じデータベースにアクセスすると、深刻なデータベース破壊を引き起こします。詳しくは、<DERBY_INSTALL_HOME>\docs\pdf\devguideにあるDerby Dev Guide (derbydev.pdf) を見てください。

Tomcat の中でデータソースをコンフィギュレーションする

次のステップは、Tomcat の中でデータソースをコンフィギュレーションすることです。Derby用にどのような環境を選択したかによって、次のような JDBC ドライバーを使用する必要があります。

- Derby がアプリケーションと同じ JVM の中で実行する埋め込み環境では、org.apache.derby.jdbc.EmbeddedDriver を使います。
- クライアント/サーバー環境を設定するネットワーク・サーバー環境では、org.apache.derby.jdbc.ClientDriver を使います。

この記事の例では、埋め込みのドライバーを使用しています。<DERBY_INSTALL_HOME>\libの中にある derby.jar ファイルには、org.apache.derby.jdbc.EmbeddedDriver クラスが含まれています。この JAR ファイルを \$CATALINA_HOME\common\libパスにコピーします。\$CATALINA_HOME\conf\server.xml を編集し、リスト 3に示すエントリーを追加します。

リスト 3. server.xml へのエントリー

```
<Context path="" docBase="ROOT" debug="0">
  <Resource name="jdbc/derbyds" auth="Container" type="javax.sql.DataSource"
    maxActive="100" maxIdle="3" maxWait="10000" username="user1" password="user1"
    driverClassName="org.apache.derby.jdbc.EmbeddedDriver"
    url="jdbc:derby:C:\\Derby\\databases\\ITEMSDB"/>
</Context>
```

次に、JSTL 宣言と、対応する SQL タグを JSP に追加する必要があります。単純に次のコード・スニペットをJSP に追加すれば、JSTL 1.1 アプリケーションは実行準備完了です。

リスト 4. 下記のコード・スニペットを追加する

```
<%@ taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql" %>
  <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
  -----
  -----
  <sql:setDataSource dataSource="jdbc/derbyds"/>
  <sql:query var="entries" sql="<sql query>"/>
  </sql:query>
```

JSTL 1.1 リリースの主なゴールは、JSTL 仕様と JSP 2.0 仕様とを同期させることです。そのため、JSTL 1.1 の taglib URI には追加の「jsp」宣言がありますが、JSTL 1.0 の taglibURI にはありません。これをリスト 5 に示します。

リスト 5. JSTL 1.0 での URI 宣言

```
<%@ taglib prefix="sql" uri="http://java.sun.com/jstl/sql" %>
  <%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>
```

行った変更を反映させるためには、Tomcat サーバーを再起動する必要があります。Tomcatの Services パネルを使うと、サーバーの起動と停止を行うことができます。

JSTL の SQL アクションを使用する

これで、すべてのコンフィギュレーションが整いましたので、いよいよ本格的に始めましょう。今度は、単純なSQL タグを使ってデータベース・クエリーと更新を行い、また複数のデータベース操作を1つのトランザクションとしてグループ化する方法について説明しましょう。

データベース・クエリー (SELECT)

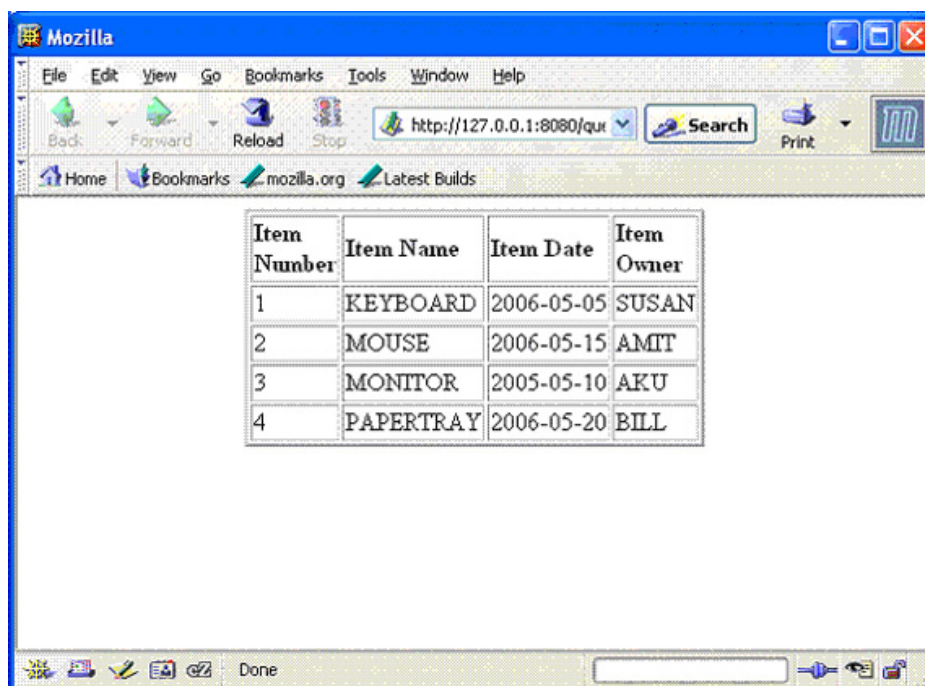
リスト 6 では、`<sql:query>` タグを使ってデータベースをクエリーしています。

リスト6. JSTL タグを使って ITEMDESC_TBL の内容を出力する

```
<sql:setDataSource dataSource="jdbc/derbyds"/>
<sql:query var="items" sql="SELECT * FROM PROJECTSCHEMA.ITEMDESC_TBL">
</sql:query>
<table border="1" align="center" valign="center">
<c:forEach var="row" items="${items.rows}">
<tr>
<td><c:out value="${row.itemnumber}"/></td>
<td><c:out value="${row.itemname}"/></td>
<td><c:out value="${row.itemdate}"/></td>
<td><c:out value="${row.itemowner}"/></td>
</c:forEach>
</table>
```

図2 は、このクエリーの結果を示したものです。

図 2. クエリーの結果のスクリーン・ショット



Item Number	Item Name	Item Date	Item Owner
1	KEYBOARD	2006-05-05	SUSAN
2	MOUSE	2006-05-15	AMIT
3	MONITOR	2005-05-10	AKU
4	PAPERTRAY	2006-05-20	BILL

リスト 6 のコードを、JSTL を使用しないリスト 7 のコードと比べてみてください。これを見ると、JSTL のタグの使いやすさがわかるでしょう。

リスト 7. Java コードを使って ITEMDESC_TBL の内容を実出力する

```
<% Context initctx=new InitialContext();
Context envCtx=(Context)initctx.lookup("java:comp/env");
javax.sql.DataSource ds=(javax.sql.DataSource)envCtx.lookup("jdbc/derbyds");
Connection conn = ds.getConnection();
Statement stmt = null;
stmt = conn.createStatement();
java.sql.ResultSet result =
    stmt.executeQuery("SELECT * FROM PROJECTSCHEMA.ITEMDESC_TBL ");
    java.sql.ResultSetMetaData metadata = result.getMetaData();
        int colCount = metadata.getColumnCount();
        int noRows=0;
            while (result.next()) {
                noRows++;
                %>
                <%
                for (int i = 1; i <= colCount; i++) {
                    %>
                    <%= result.getString(i) %>
                    <%
                }
                %>
                <%
            }
        conn.close(); %>
```

データベースの更新 (INSERT、UPDATE、または DELETE)

<sql:update> タグは、SQL の、INSERT、UPDATE、または DELETE 文を実行します(リスト 8 から 10)。また、何も返さない SQL 文 (SQL DDL 文など) を実行することもできます。

リスト 8. <sql:update> タグを使った SQL 挿入

```
<sql:update dataSource="jdbc/derbyds">
INSERT INTO PROJECTSCHEMA.ITEMDESC_TBL (ITEMNUMBER, ITEMNAME, ITEMOWNER)
VALUES(005 ,? ,?)
<sql:param value="iPod"/>
<sql:param value="AMY"/>
</sql:update>
```

リスト 9. <sql:update> タグを使った SQL 更新

```
<sql:setDataSource dataSource="jdbc/derbyds"/>
<sql:update>
UPDATE PROJECTSCHEMA.ITEMDESC_TBL
SET ITEMNAME = ?
WHERE ITEMNUMBER = ?
<sql:param value="PENHOLDER"/>
<sql:param value="1"/>
</sql:update>
```

リスト 10. <sql:update> タグを使った SQL 削除

```
<sql:update dataSource="jdbc/db2ds">
DELETE FROM PROJECTSCHEMA.ITEMDESC_TBL
WHERE ITEMNUMBER = ?
<sql:param value="3"/></sql:update>
```


1つのトランザクションで複数のデータベースを操作する

JSTL のデータベース・タグ・ライブラリーの最後のコンポーネントは、`<sql:transaction>` タグです。クエリー操作と更新操作を1つのトランザクションとしてグループ化することによって、操作は必ず、すべて成功か、あるいは(ある1つの操作が失敗した場合に)すべてロールバックされるかのどちらかになります。1つのトランザクション内での操作は、すべて成功か、すべて失敗のどちらかになるのです。

許可されている JDBC トランザクション・レベルは下記の通りです。

- `none`
- `read_committed`
- `read_uncommitted`
- `repeatable_read`
- `serializable`

デフォルトでは、接続のトランザクション分離レベルの設定がそのまま維持されます。分離レベルはデータベースとデータソースの実装によって異なります。

リスト 11 では、`<sql:transaction>` アクションが `<sql:query>` サブタグと `<sql:update>` サブタグに対するトランザクション・コンテキストを確立しています。

リスト 11. `<sql:transaction>` を使ってトランザクションを作成する

```
<sql:transaction>
<sql:update>UPDATE PROJECTSCHEMA.ITEMDESC_TBL
SET ITEMNAME = ?
WHERE ITEMNUMBER = ?
  <sql:param value="MM"/>
  <sql:param value="2"/>
</sql:update>
<sql:update>UPDATE PROJECTSCHEMA.ITEMDESC_TBL
SET ITEMOWNER = ?
WHERE ITEMNUMBER = ?
  <sql:param value="JOHN"/>
  <sql:param value="2"/>
</sql:update>
</sql:transaction>
```

まとめ

以上で、JSTL 1.1 の SQL タグ・ライブラリーと Tomcat 5.5、そして ApacheDerby リレーショナル・データベースを使って、データベース・クエリーを本来あるべき程度に容易にする方法がわかりました。JSTL 1.1 のメンテナンス・リリースには Servlet 2.4 仕様と JSP 2.0 仕様をサポートする JSP コンテナが必要なため、JSTL ライブラリーは Tomcat 5.5 の中でコンフィギュレーションします。この記事では、ApacheDerby データベースやスキーマ、テーブルなどを、ij (Interactive Java) ツールを使って容易に設定できることも学びました。また、Tomcat 5.5 で使われている JDBC ドライバーを学ぶと共に、クエリーや更新のための単純な SQL アクションの使い方、いくつかの操作を1つのトランザクションにグループ化する方法についても学びました。

JSTL の最新リリースである JSTL 1.2 には JSP 2.1 コンテナが必要なこと、また JSTL 1.2 以降では JSTL が Java EE の一部を構成することに注意してください。JSTL 1.2 リリースでの変更に関して詳しくは、[参考文献](#)を見てください。

ダウンロード可能なリソース

内容	ファイル名	サイズ
JSPs with JSTL SQL Tags	j-jstlsql_source.zip	12KB

関連トピック

- [「JSTL入門: 式言語」](#) (Mark Kolb 著、developerWorks、2003年2月) は、JSTL とSQL クエリーについて学ぶための出発点です。
- [「Integrate Cloudscape Version 10 or Derby with Tomcat」](#) (Lance Bader 著、developerWorks、2005年8月) は、Cloudscape または Derby と Tomcat との統合方法として考えられるシナリオを解説しています。
- [「Use Derby in a J2EE Server environment」](#) (Stanley Bradbury 著、developerWorks、2006年6月) を読んで、Cloudscape または Derby と J2EE 開発環境とを統合する方法について学んでください。
- [JSTL の仕様](#)を見て、JSTL の SQL アクションについて学んでください。
- [Tomcat のホームページ](#)には、JDBC リソースのコンフィギュレーションに関する Tomcat のドキュメンテーションがあります。
- [Apache Derby](#)にあるマニュアルを見て、この一般的なリレーショナル・データベース管理システムについて学んでください。
- [Java technology ゾーン](#)には、Java プログラミングに関するあらゆる話題を網羅した記事が豊富に用意されています。
- [Tomcat 5.5](#) のバイナリー・ディストリビューションをダウンロードしてください。
- この記事で取り上げたリレーショナル・データベース管理システム、[Apache Derby](#) をダウンロードしてください。
- JSTL and SQL のために、最新の JSTL ビルドを入手してください。

© Copyright IBM Corporation 2006

(www.ibm.com/legal/copytrade.shtml)

商標

(www.ibm.com/developerworks/jp/ibm/trademarks/)