Adaptive Server Anywhere 9 で Hibernate を使用する

本書は、Adaptive Server Anywhere を使用していて、開発を容易にするための新しいテクノロジを検討している Java 開発者を対象としています。本書では、Hibernate を使用して Java オブジェクトをデータベース・テーブルにマップする方法について説明します。

Hibernate とは？

Hibernate は、XML 設定ファイルを使用して Java オブジェクトをデータベース・テーブルにマップするオープンソース・オブジェクト／リレーショナル・マッピング(ORM)・ソリューションです。また、Hibernate は、データベースから Java オブジェクトへのレコードの抽出を可能にする独自のクエリ言語(HQL)も備えています。Hibernate は、数多くの一般的なデータベース(例えば DB2, Sybase, Adaptive Server Anywhere, and MySQL)を操作します。Hibernate では、特定の JDBC ドライバまたは JDK に対する要件は明示的に指定されていませんが、JDBC 2.0 に対応している JDBC ドライバを JDK 1.4 以上と共に使用することを推奨しています。

Hibernate の利点

Hibernate は、JDBC および SQL の直接操作から開発者を解放します。したがって、Java プログラマは、オブジェクトの操作に携わるだけでよく、データの持続性について心配する必要はありません。Hibernate では、ユーザはマッピング XML ドキュメントを作成し、どのように Java オブジェクト(JavaBean)がデータベース・テーブルにマップされるのかを表す必要があります。Java オブジェクトの属性は、データベース内の該当するカラムに対応しています。また、データベースに接続する場合に必要な接続パラメータを格納するために、*hibernate.cfg.xml* 設定ファイルがセットアップされます。すべての設定ファイルを設定したら、Session.save (*object*)メソッドを使用して Java オブジェクトを作成し、保存できます。これにより、データベースに対する適切な SQL 文が生成され、その文がデータベースに保持されます。また、Hibernate は、非常に便利な以下の 2 つのツールも提供します。

- SchemaExport.*hbm.xml* 設定ファイルの設定に基づいて、データベースに接続し、適切なスキーマ (DDL)を生成する
- HBM2JavaTask.*hbm.xml* 設定ファイルの設定に基づいて、標準 JavaBean クラス(POJO-plain old java object)を生成する

開発者は JDBC または SQL を操作する必要がないため、Hibernate で開発されたアプリケーションは、基本となるデータベース・スキーマで行われた変更に対する回復機能が向上します。また、開発者は選択したデータベースに依存する特殊な考慮事項を持った取得済み結果セットを手動で処理する必要がないため、移植性も向上します。さらに、Hibernate により開発者は、操作される Java オブジェクトとデータベース内の継続データの間で一貫した状態を維持することができます。プログラム・スペース内でオブジェクトに加えられた変更は、データベースに反映されます。要約すると、Hibernate の主な利点は、アプリケーションの継続的側面の抽象化とその処理の簡略化によって開発時間を短縮するということです。

Adaptive Server Anywhere で Hibernate を使用する (イントロダクション)

このセクションでは、Adaptive Server Anywhere データベースへ Java オブジェクトの保存と読み込みを行う方法について紹介します。

Adaptive Server Anywhere へ入出力するオブジェクトの保存と読み込み

Adaptive Server Anywhere で Hibernate を使う 7 つのステップ

1. データベースを初期化する。
2. データベース・スキーマを準備する。
3. Java オブジェクトを記述する。
4. データベースのテーブルと Java オブジェクトのマッピングを XML ファイルに記述する。
5. 接続のために hibernate.hbm.xml 設定ファイルを準備する。
6. データベースと入出力を行うオブジェクトの保存と読み込みのために HibernateManager を記述する。
7. ファイルのコンパイルとビルドを行う。

ステップ 1: データベースを初期化する

1. コマンド・プロンプトを開く。
2. test.db という名前のデータベースを作成するために、以下のコマンドを実行する。

   dbinit test.db

3. test.db データ・ベースを実行するデータ・ベース・サーバーを起動する。

   dbeng9 test.db

4. Interactive SQL から test.db データベースに接続する。

   dbisql -c "UID=dba;PWD=sql;ENG=test"

ステップ 2: データベース・スキーマを準備する

1. Book という名前のテーブルをデータベースに追加する

```
CREATE TABLE "DBA"."book"
(
   "book_id"      INTEGER NOT NULL DEFAULT AUTOINCREMENT,
   "TITLE"        VARCHAR(255) NULL,
   "PRICE"        DOUBLE NULL,
    PRIMARY KEY ("book_id"),
)
go
COMMIT WORK
      go
```

ステップ 3:ミューテータとアクセサを使用して単純な Book オブジェクトを作成する

1. Book テーブルをモデル化する Plain Old Java Object(POJO) を作成します。Hibernate を操作する Java オブジェクトはデフォルトのコンストラクタを持っている必要があることに注意してください。

```
public class Book {

      private int book_id;
      private String title;
      private double price;


      public Book() {}

      public int getBook_id() {
      return book_id;
      }

      public void setBook_id(int num) {
      this.book_id = num;
      }
      public String getTitle() {
      return title;
      }

      public void setTitle(String name) {
      this.title = name;
      }

      public double getPrice() {
      return price;
      }

      public void setPrice(double cost) {
      this.price = cost;
      }

}
```

ステップ 4: Java オブジェクトをデータベース・テーブルにマップする

1. Book Java オブジェクトを Book データベース・テーブルと結びつける Book.hbm.xml 構成ファイルを作成します。

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>

  <class name="Book" table="BOOK">

        <id name="book_id" column="book_id" >
         <generator class="native"/>
        </id>

        <property name="title">
             <column name="TITLE" />
        </property>

        <property name="price">
            <column name="PRICE"/>
        </property>
   </class>

</hibernate-mapping>
```

- クラス名タグでは、マップしている Java オブジェクト・クラスの場所が指定されます。また、テーブル属性では、Java オブジェクトのマップ先となるテーブルが指定されます。
- ID 要素は、プライマリ・キー用です。generate class=native を指定すると、使用されるデータベースに基づいて（この場合は、オートインクリメントに基づいて）プライマリ・キーが生成されることを意味します。オプションとして、generator class="assigned" を設定し、アプリケーションでプライマリ・キー ID を手動設定することができます。
- プロパティ要素では、Java オブジェクト(book) の各属性がどのようにデータベース内の特定のカラムにマップされるのかが指定されます。この場合は、title 属性をデータベース内の title カラムにマップし、price 属性を price カラムにマップしています。

ステップ 5: Adaptive Server Anywhere データベースに接続するための hibernate.cfg.xml ファイルを設定する

1. *hibernate.cfg.xml* ファイルを設定します。このファイルは、JDBC を使用して Adaptive Server Anywhere データベースに接続する方法を Hibernate に伝えます。この例では、データベース test.db に接続します。

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
"-//Hibernate/Hibernate Configuration DTD//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
   <session-factory>

     <property
     name="hibernate.connection.driver_class">com.sybase.jdbc2.jdbc.S
     ybDriver</property>


     <property
     name="hibernate.connection.url">jdbc:sybase:Tds:localhost:2638</
     property>

     <property name="hibernate.connection.username">dba</property>

     <property name="hibernate.connection.password">sql</property>

     <property name="hibernate.connection.pool_size">10</property>

     <property name="show_sql">true</property>

     <property
     name="dialect">org.hibernate.dialect.SybaseAnywhereDialect</prop
     erty>

     <mapping resource="Book.hbm.xml"/>
   </session-factory>
</hibernate-configuration>
```

- Driver_class **では、**JDBC **ドライバが指定されます。この例では** iAnywhere JDBC **ドライバを使用しますが、**jConnect **ドライバを使用することも可能です。以下の例は、**jConnect **ドライバを使用してサンプルの** Adaptive Server Anywhere **データベースに接続する方法を示しています。**

  <property
  name="hibernate.connection.driver_class">com.sybase.jdbc2.jdbc.SybDriver</property>

- **データベースのユーザ** ID **とパスワードを指定する必要があります。**
- **ダイアレクトは、**HQL **からベンダ固有の** SQL **に変換する方法を** Hibernate **に指示します。**Hibernate **の開発者は、**Adaptive Server Anywhere 8 **用のダイアレクト・ファイルを提供しています。このファイルは、バージョン** 9 **でも機能します。興味がある場合は、**Hibernate **ソース・コードを調べて、どのデータ型がサポートされているのかを確認できます。**
- Mapping Resource **データベースにマップされている** Java **オブジェクトのどれが接続に関係しているのかを示します。複数のオブジェクト・マッピングが存在する場合は、複数の** .hbm.xml **ファイルを含めてください。**

ステップ 6: データベースへ入出力を行う 作成、保存、読み込み オブジェクトのために UtilManager を記述する

以下は、簡単な UtilManager **の例です**:

```
import org.hibernate.*;
import org.hibernate.cfg.*;

public class UtilManager {

	public static void main (String[] args) {
		Session session = null;

		try {

			/*we create a SessionFactory based on the
			hibernate.cfg.xml file, from a SessionFactory, we can
			create an individual Session*/

			SessionFactory sessionfactory = new
			Configuration().configure().buildSessionFactory();
			session = sessionfactory.openSession();

			System.out.println("Inserting Record");

			//Creating Book Object
			Book book = new Book();

			//Begin a transaction within the session
			Transaction tx = session.beginTransaction();

			book.setTitle("Green Eggs and Ham");
			book.setPrice(5.99);

			//save the book object into the database
			session.save(book);

			//the change in memory would not propagate to
			//database unless a transaction is committed or
			//rollback.
			tx.commit();
			session.close();

			System.out.println("Done");
		} catch(Exception e) {
			System.out.println(e.getMessage());
		}
	}
}
```

ステップ 7: ファイルのコンパイルとビルドを行う

注意: Java **コンパイラ** (javac) **で以下の命令を使用する代わりに、**Apache **の** ANT **を使用することも可能です。詳細については以下を参照してください。**http://www.hibernate.org/hib_docs/v3/reference/en/html/tutorial.html.

1. *Book.java* **をコンパイルします。**

javac Book.java

2. **インクルードする** CLASSPATH **を設定します。**
   a. JDBC **ドライバー** (*jconn2.jar* **または** *jodbc.jar*). **デフォルトの場所は** *C:¥Program Files¥Sybase¥Shared¥jconnect-5_5¥classes¥jconn2.jar*. Hibernate *.jar* files (*Hibernate¥lib* **フォルダに保存されている。**).
   以下のリストは Hibernate **が** C:¥Program Files¥hibernate3 **にインストールされていると仮定しています。**

      C:¥Program Files¥hibernate3¥hibernate3.jar
      C:¥Program Files¥hibernate3¥lib¥dom4j-1.6.1.jar
      C:¥Program Files¥hibernate3¥lib¥commons-logging-1.0.4.jar
      C:¥Program Files¥hibernate3¥lib¥commons-collections-2.1.1.jar
      C:¥Program Files¥hibernate3¥lib¥ehcache-1.1.jar
      C:¥Program Files¥hibernate3¥lib¥jta.jar
      C:¥Program Files¥hibernate3¥lib¥cglib-2.1.3.jar
      C:¥Program Files¥hibernate3¥lib¥asm.jar
      C:¥Program Files¥hibernate3¥lib¥antlr-2.7.6rc1.jar

   b. *.xml* **ファイルを設定する**
   c. *Book.class* **ファイル**

   例 CLASSPATH **は以下のように見えます**

   .;C:¥Program Files¥Sybase¥Shared¥jconnect-5_5¥classes¥jconn2.jar;C:¥Program Files¥Sybase¥Shared¥jconnect-5_5¥classes;C:¥Program Files¥Java¥jdk1.5.0_06¥lib;C:¥Program Files¥hibernate3¥hibernate3.jar;C:¥Program Files¥hibernate3¥lib¥dom4j-1.6.1.jar;C:¥Program Files¥hibernate3¥lib¥commons-logging-1.0.4.jar;C:¥Program Files¥hibernate3¥lib¥commons-collections-2.1.1.jar;C:¥Program Files¥hibernate3¥lib¥ehcache-1.1.jar;C:¥Program Files¥hibernate3¥lib¥jta.jar;C:¥Program Files¥hibernate3¥lib¥cglib-2.1.3.jar;C:¥Program Files¥hibernate3¥lib¥asm.jar;C:¥Program Files¥hibernate3¥lib¥antlr-2.7.6rc1.jar

3. *UtilManager.java* **をコンパイルします。**

   javac UtilManager.java

4. UtilManager **を実行して、**Adaptive Server Anywhere **データベースで結果を確認します。**(**例**: using Sybase

Central)

java UtilManager

レコードをオブジェクトにロードする

Java オブジェクトをデータベースに保存できるようになったので、ユーザはデータベース・レコードを Java オブジェクトから取得したいと考える可能性があります。

以下のコードを UtilManager に追加した場合は、HQL(hibernate query language) を使用してデータベースから結果セットを取得し、その結果セットをオブジェクトのリストに自動的にロードすることができます。この例は、結果セット内の最初のオブジェクトのタイトルを検索します。createQuery ("from Book")では Book テーブルの代わりに Book クラスの問い合わせが行われていることに注目してください。

```
import org.hibernate.*;
import org.hibernate.cfg.*;
import java.util.List;

public class UtilManager {

      public static void main (String[] args) {
            Session session = null;

            try {

                  /*we create a SessionFactory based on the
                  hibernate.cfg.xml file, from a SessionFactory, we can
                  create an individual Session*/

                  SessionFactory sessionfactory = new
                  Configuration().configure().buildSessionFactory();
                  session = sessionfactory.openSession();

                  System.out.println("Inserting Record");

                  //Creating Book Object
                  Book book = new Book();

                  //Begin a transaction within the session
                  Transaction tx = session.beginTransaction();

                  book.setTitle("Green Eggs and Ham");
                  book.setPrice(5.99);

                  //save the book object into the database
                  session.save(book);

                  //the change in memory would not propagate to
                  //database unless a transaction is committed or
                  //rollback.
                  tx.commit();

                  // Perform a query
                  List books = session.createQuery("from Book").list();

                  if (books.isEmpty() != true) {
                        System.out.println ("First Book Exists - its
                        title is:");
                        Book firstbook = (Book)books.get(0);
                        System.out.println (firstbook.getTitle());
                  } else {
                        System.out.println ("List is Empty");
                  }

                  session.close();

                  System.out.println("Done");
            } catch(Exception e) {
                  System.out.println(e.getMessage());
            }
      }
```

まとめ

Hibernate を操作できるツールは数多くあります。たとえば、Ant はビルド・プロセスで日常的に Hibernate と共に使用されます。また、SchemaExport および HBM2JavaTask ツールを使用してプロセス全体を単純化することもできます。

Hibernate には、ほかにも 1 対多の関連付け、多対多の関連付け、継承、データベースに対する Java オブジェクトの多様性マッピング (polymorphism mapping) といった数多くの機能があります。ただし、これらのトピックには、高度な Hibernate の知識が必要となります。最初は、Hibernate Web サイトの Hibernate チュートリアル・ガイドに目を通すことをお勧めします。

関連情報

1. Hibernate main page
   http://www.hibernate.org
2. Hibernate manual
   http://www.hibernate.org/hib_docs/v3/reference/en/html/index.html
3. Hibernate introduction using a stand-alone application
   http://www.hibernate.org/hib_docs/v3/reference/en/html/tutorial.html
4. iAnywhere Solutions 株式会社 Web サイト
   http://www.ianywhere.jp/
5. iAnywhere デベロッパー・コミュニティ・ページ
   http://www.ianywhere.jp/developers/index.html
6. iAnywhere ダウンロード・ページ (SQL Anywhere Developer Edition やその他のリソースを入手できます)
   http://www.ianywhere.jp/dl/index.html