Глава 20

JDBC

Драйверы, соединения и запросы

JDBC (Java DataBase Connectivity) – стандартный прикладной интерфейс (API) языка Java для организации взаимодействия между приложением и СУБД. Это взаимодействие осуществляется с помощью драйверов JDBC, обеспечивающих реализацию общих интерфейсов для конкретных СУБД и конкретных протоколов. В JDBC определяются четыре типа драйверов:

1. Драйвер, использующий другой прикладной интерфейс взаимодействия с СУБД, в частности ODBC (так называемый JDBC-ODBC – мост). Стандартный драйвер первого типа **`sun.jdbc.odbc.JdbcOdbcDriver`** входит в JDK.
2. Драйвер, работающий через внешние (native) библиотеки (т.е. клиента СУБД).
3. Драйвер, работающий по сетевому и независимому от СУБД протоколу с промежуточным Java-сервером, который, в свою очередь, подключается к нужной СУБД.
4. Сетевой драйвер, работающий напрямую с нужной СУБД и не требующий установки native-библиотек.

Предпочтение естественным образом отдается второму типу, однако если приложение выполняется на машине, на которой не предполагается установка клиента СУБД, то выбор производится между третьим и четвертым типами. Причем четвертый тип работает напрямую с СУБД по ее протоколу, поэтому можно предположить, что драйвер четвертого типа будет более эффективным по сравнению с третьим типом с точки зрения производительности. Первый же тип, как правило, используется редко, т.е. в тех случаях, когда у СУБД нет своего драйвера JDBC, зато есть драйвер ODBC.

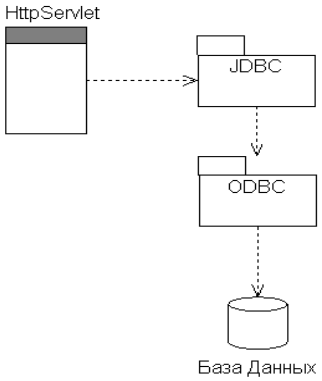

Рис. 20.1. Доступ к БД с помощью JDBC-ODBC-моста

JDBC предоставляет интерфейс для разработчиков, использующих различные СУБД. С помощью JDBC отсылаются SQL-запросы только к реляционным базам данных (БД), для которых существуют драйверы, знающие способ общения с реальным сервером базы данных.

Строго говоря, JDBC не имеет прямого отношения к J2EE, но так как взаимодействие с СУБД является неотъемлемой частью Web-приложений, то эта технология рассматривается в данном контексте.

Последовательность действий

1. *Загрузка класса драйвера базы данных при отсутствии экземпляра этого класса.*

Например:

```
String driverName = "org.gjt.mm.mysql.Driver";
```

для СУБД MySQL,

```
String driverName = "sun.jdbc.odbc.JdbcOdbcDriver";
```

для СУБД MSAccess или

```
String driverName = "org.postgreesql.Driver";
```

для СУБД PostgreeSQL.

После этого выполняется собственно загрузка драйвера в память:

```
Class.forName(driverName);
```

и становится возможным соединение с СУБД.

Эти же действия можно выполнить, импортируя библиотеку и создавая объект явно. Например, для СУБД DB2 от IBM объект-драйвер можно создать следующим образом:

```
new com.ibm.db2.jdbc.net.DB2Driver();
```

2. *Установка соединения с БД.*

Для установки соединения с БД вызывается статический метод **`getConnection()`** класса **`DriverManager`**. В качестве параметров методу передаются URL базы данных, логин пользователя БД и пароль доступа. Метод возвращает объект **`Connection`**. URL базы данных, состоящий из типа и адреса физического расположения БД, может создаваться в виде отдельной строки или извлекаться из файла ресурсов. Например:

```
Connection cn = DriverManager.getConnection(
"jdbc:mysql://localhost/my_db", "root", "pass");
```

В результате будет возвращен объект **`Connection`** и будет одно установленное соединение с БД **`my_db`**. Класс **`DriverManager`** предоставляет средства для управления набором драйверов баз данных. С помощью метода **`registerDriver()`** драйверы регистрируются, а методом **`getDrivers()`** можно получить список всех драйверов.

3. *Создание объекта для передачи запросов.*

После создания объекта **`Connection`** и установки соединения можно начинать работу с БД с помощью операторов SQL. Для выполнения запросов применяется объект **`Statement`**, создаваемый вызовом метода **`createStatement()`** класса **`Connection`**.

```
Statement st = cn.createStatement();
```

Объект класса **Statement** используется для выполнения SQL-запроса без его предварительной подготовки. Могут применяться также объекты классов **PreparedStatement** и **CallableStatement** для выполнения подготовленных запросов и хранимых процедур. Созданные объекты можно использовать для выполнения запроса SQL, передавая его в один из методов **executeQuery(String sql)** или **executeUpdate(String sql)**.

4. *Выполнение запроса.*

Результаты выполнения запроса помещаются в объект **ResultSet**:

```
ResultSet rs = st.executeQuery(
"SELECT * FROM my_table");//выборка всех данных таблицы my_table
```

Для добавления, удаления или изменения информации в таблице вместо метода **executeQuery()** запрос помещается в метод **executeUpdate()**.

5. *Обработка результатов выполнения запроса* производится методами интерфейса **ResultSet**, где самыми распространенными являются **next()** и **getString(int pos)** а также аналогичные методы, начинающиеся с **get**Тип**(int pos)** (**getInt(int pos)**, **getFloat(int pos)** и др.) и **update**Тип**()**. Среди них следует выделить методы **getClob(int pos)** и **getBlob(int pos)**, позволяющие извлекать из полей таблицы специфические объекты (Character Large Object, Binary Large Object), которые могут быть, например, графическими или архивными файлами. Эффективным способом извлечения значения поля из таблицы ответа является обращение к этому полю по его позиции в строке.

При первом вызове метода **next()** указатель перемещается на таблицу результатов выборки в позицию первой строки таблицы ответа. Когда строки закончатся, метод возвратит значение **false**.

6. *Закрытие соединения*

```
cn.close();
```

После того как база больше не нужна, соединение закрывается.

Для того чтобы правильно пользоваться приведенными методами, программисту требуется знать типы полей БД. В распределенных системах это знание предполагается изначально.

СУБД MySQL

СУБД MySQL совместима с JDBC и будет применяться для создания экспериментальных БД. Последняя версия СУБД может быть загружена с сайта **www.mysql.com**. Для корректной установки необходимо следовать инструкциям мастера установки. Каталог лучше выбирать по умолчанию. В процессе установки следует создать администратора СУБД с именем `root` и паролем `pass`. Если планируется разворачивать реально работающее приложение, необходимо исключить тривиальных пользователей сервера БД (иначе злоумышленники могут получить полный доступ к БД). Для запуска следует использовать команду из папки **/mysql/bin**:

```
mysqld-nt -standalone
```

Если не появится сообщение об ошибке, то СУБД MySQL запущена. Для создания БД и таблиц используются команды языка SQL.