Пул соединений

При большом количестве клиентов, работающих с приложением, к его базе данных выполняется большое количество запросов. Установление соединения с БД является дорогостоящей (по требуемым ресурсам) операцией. Эффективным способом решения данной проблемы является организация пула (pool) используемых соединений, которые не закрываются физически, а хранятся в очереди и предоставляются повторно для других запросов.

Пул соединений – это одна из стратегий предоставления соединений приложению (не единственная, да и самих стратегий организации пула существует несколько).

Пул соединений можно организовать с помощью **server.xml** дескрипторного файла Apache Tomcat в виде:

```
<Context docBase="FirstProject" path="/FirstProject"
reloadable="true" source="com.ibm.etools.webtools.server:
FirstProject">
<!—создание пул соединений для СУБД MySQL -->
<Resource auth="Container" name="jdbc/db1"
                           type="javax.sql.DataSource"/>
<ResourceParams name="jdbc/db1">
      <parameter>
            <name>factory</name>
      <val-
ue>org.apache.commons.dbcp.BasicDataSourceFactory</value>
      </parameter>
      <parameter>
            <name>driverClassName</name>
            <value>org.gjt.mm.mysql.Driver</value>
      </parameter>
<!—url-адрес соединения JDBC для конкретной базы данных db1
Аргумент autoReconnect=true, заданный для url-адреса драйвера JDBC, должен
автоматически разорвать соединение, если mysqld его закроет. По умолчанию
mysqld закроет соединение через 8 часов.-->
      <parameter>
            <name>url</name>
<value>jdbc:mysql://localhost:3306/db1?autoReconnect=true
</value>
      </parameter>
      <parameter>
            <name>username</name>
            <value>root</value>
      </parameter>
      <parameter>
            <name>password</name>
            <value>pass</value>
      </parameter>
      <parameter>
            <name>maxActive</name>
```

```
            <value>500</value>
        </parameter>
        <parameter>
            <name>maxIdle</name>
            <value>10</value>
        </parameter>
        <parameter>
            <name>maxWait</name>
            <value>10000</value>
        </parameter>
        <parameter>
            <name>removeAbandoned</name>
            <value>true</value>
        </parameter>
        <parameter>
            <name>removeAbandonedTimeout</name>
            <value>60</value>
        </parameter>
        <parameter>
            <name>logAbandoned</name>
            <value>true</value>
        </parameter>
</ResourceParams>
</Context>
```

Найти и запустить данный пул соединений можно с помощью JNDI.

Разделяемый доступ к источнику данных можно организовать, например, путем объявления статической переменной типа **DataSource** из пакета **javax.sql**, однако в J2EE принято использовать для этих целей каталог. Источник данных типа **DataSource** – это компонент, предоставляющий соединение с приложением СУБД.

Класс **InitialContext**, как часть JNDI API, обеспечивает работу с каталогом именованных объектов. В этом каталоге можно связать объект источника данных **DataSource** с некоторым именем (не только с именем БД, но и вообще с любым), предварительно создав объект **DataSource**.

Затем созданный объект можно получить с помощью метода **lookup()** по его имени. Методу **lookup()** передается имя, всегда начинающееся с имени корневого контекста.

```
javax.naming.Context ct =
   new javax.naming.InitialContext();
DataSource ds = (DataSource)ct.lookup("java:jdbc/db1");
Connection cn = ds.getConnection("root", "pass");
```

После выполнения запроса соединение завершается, и его объект возвращается обратно в пул вызовом:

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

Некоторые производители СУБД для облегчения создания пула соединений определяют собственный класс на основе интерфейса **DataSource**. В этом случае пул соединений может быть создан, например, следующим образом:

```
import COM.ibm.db2.jdbc.DB2DataSource;
      ...
      DB2DataSource ds = new DB2DataSource();
      ds.setServerName("//localhost:50000/db1");
      Connection cn = ds.getConnection("db2inst1", "pass");
```

Драйвер определяется автоматически в объекте **`DB2DataSource`**.

Задания к главе 20

Вариант А

В каждом из заданий необходимо выполнить следующие действия:

- Организацию соединения с базой данных вынести в отдельный класс, метод которого возвращает соединение.
- Создать БД. Привести таблицы к одной из нормированных форм.
- Создать класс для выполнения запросов на извлечение информации из БД с использованием компилированных запросов.
- Создать класс на добавление информации.
- Создать HTML-документ с полями для формирования запроса.
- Результаты выполнения запроса передать клиенту в виде HTML-документа.

1. **Файловая система.** В БД хранится информация о дереве каталогов файловой системы – каталоги, подкаталоги, файлы.
   Для каталогов необходимо хранить:
   – родительский каталог;
   – название.
   Для файлов необходимо хранить:
   – родительский каталог;
   – название;
   – место, занимаемое на диске.
   - Определить полный путь заданного файла (каталога).
   - Подсчитать количество файлов в заданном каталоге, включая вложенные файлы и каталоги.
   - Подсчитать место, занимаемое на диске содержимым заданного каталога.
   - Найти в базе файлы по заданной маске с выдачей полного пути.
   - Переместить файлы и подкаталоги из одного каталога в другой.
   - Удалить файлы и каталоги заданного каталога.
2. **Видеотека.** В БД хранится информация о домашней видеотеке – фильмы, актеры, режиссеры.
   Для фильмов необходимо хранить:
   – название;
   – имена актеров;
   – дату выхода;
   – страну, в которой выпущен фильм.
   Для актеров и режиссеров необходимо хранить:
   – ФИО;
   – дату рождения.
   - Найти все фильмы, вышедшие на экран в текущем и прошлом году.