УДК 800.92

РАЗРАБОТКА ИНТЕРНЕТ-МАГАЗИНА С ПРИМЕНЕНИЕМ OST-ТЕХНОЛОГИЙ

Р.А.КУКАНОВА, Г.Ж.ЭРКИНБАЕВА
E.mail. ksucta@elcat.kg

Бул статьяда Oracle тилинде базаны түзүү жана Java жогорку деңгеелдеги тилинде интерфейс менен байланыштыруу, ошондой эле Database Connection Pool артыкчылыктарын колдонуу каралган.

В статье рассмотрены создание и связывание базы Oracle с интерфейсом на языке высокого уровня Java, в том числе пользование преимуществами Database Connection Pool.

We wrote this article on our scientific theme about database from Oracle and create the interface in Java, used properties Database Connection Pool.

В настоящее время трудно представить себе веб-приложение, которое не использовало бы базу данных для своих нужд. При работе с базой данных очень важно следить за соединениями к базе и вовремя освобождать их. Для этих целей разработчики веб-приложений пользуются так называемыми Connection Pools или же используют, исправляют уже существующие. Одним из лучших сервлет-контейнеров является Apache Tomcat. Tomcat хорош еще и тем, что он использует свой собственный DBCP (Database Connection Pool).
О том, как в полной мере воспользоваться преимуществами DBCP сервера Tomcat, и будет эта статья.

Предположим, в сети по IP-адресу 192.168.0.5 находится Oracle с базой данных TESTDB и пользователем DBuser с паролем DBpass. Для того чтобы настроить Tomcat на эту базу, необходимо в конфигурационном файле conf\server.xml в разделе Context, если его нет, создать его внутри раздела Host непосредственно перед его окончанием, затем добавить следующие строки:

```
<resource name="newDS" auth="Container"
type="javax.sql.DataSource"/>
<resourceparams name="newDS">
 <parameter>
  <name>factory</name>
  <value>org.apache.commons.dbcp.BasicDataSourceFactory</value>
 </parameter>
 <parameter>
  <name>url</name>
  <value>jdbc:oracle:thin:@192.168.0.10:1521:TESTDB</value>
 </parameter>
 <parameter>
  <name>username</name>
  <value>DBuser</value>
 </parameter>
 <parameter>
  <name>password</name>
  <value>DBpass</value>
 </parameter>
 <parameter>
   <name>driverClassName</name>
```

```
    <value>oracle.jdbc.driver.OracleDriver</value>
  </parameter>
  <parameter>
   <name>maxActive</name>
   <value>100</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>
 </resourceparams>
```

С помощью этой конфигурации создается Data Source с именем newDB.

Рассмотрим параметры этого объекта.

Параметр factory говорит о том, какую фабрику нужно использовать для создания объектов, реализующих интерфейс DataSource. Следующий параметр представляет собой URL для доступа к базе данных. Параметры username и password содержат соответственно имя пользователя/схемы и пароль. Имя драйвера базы данных указано в параметре driverClassName.

Следующие параметры отвечают непосредственно за Connection Pool. Рассмотрим их немного подробнее.

maxActive – максимальное количество соединений, которые будут содержаться в пуле. Если значение этого параметра установить в 0, то ограничения на количество подключений к базе не будет. В конфигурации самого сервера баз данных такой параметр тоже существует, его значение должно быть больше, чем maxActive. maxIdle – максимальное количество простаивающих соединений, которые будут оставаться в пуле. При значении этого параметра, равном нулю, ограничений не будет. maxWait – если время ожидания соединения превысит значение параметра maxWait (в миллисекундах), пользователь получит Exception. При значении maxWait, равном -1, время ожидания не ограничено. Так как Tomcat работает под виртуальной машиной (JVM), для освобождения памяти JVM использует сборщик мусора (garbage collector). GC имеет очень высокий приоритет и при освобождении памяти Tomcat будет входить в состояние ожидания (на доли секунды, реже – на несколько секунд). При маленьких значениях maxWait соединение может не установиться из-за тайм-аута. Значение 10000 в подавляющем большинстве случаев будет оптимальным. removeAbandoned – при установке этого параметра в true заброшенные соединения будут освобождены, когда количество свободных соединений в пуле невелико. removeAbandonedTimeout – этот параметр указывает в секундах время, через которое любое простаивающее соединение будет считаться заброшенным. После этого нужно настроить веб-приложение для работы с DataSource. Делается это так: в файле-дескрипторе web.xml нужно написать:

```
<resource -ref>
 <description>Описание ресурса</description>
```

```
  <res -ref-name>newDB</res>
  <res -type>javax.sql.DataSource</res>
  <res -auth>Container</res>
 </resource>
```

Чтобы иметь доступ к DataSource через JNDI и создавать соединения с базой,можно написать себе такой класс:

```
public class ConnectionFactory {
  private static final String DATASOURCE_NAME = “newDB”;
  private static DataSource ds = null;
  static {
    Context initContext = new InitialContext();
    Context envContext = (Context)initContext.lookup(“java:/comp/env”);
    ds = (DataSource) envContext.lookup(DATASOURCE_NAME);
  }
  public static Connection getConnection() throws SQLException{
    Connection conn = ds.getConnection();
    return conn;
  }
}
```

В дальнейшей работе, учитывая, что тактовая частота типичного центрального процессора и объем памяти серверов растут почти так же быстро, как и сам Интернет, упомянутое предположение является разумным для серверов, обслуживающих до нескольких тысяч запросов в секунду, в частности, еще и потому, что Java с базой хорошо работает в серверных конфигурациях с несколькими центральными процессорами.

Однако в дальнейшей работе сайта, возможно, придется предусмотреть возможность нехватки ресурсов одного сервера для того, чтобы справиться с нагрузкой. Помимо увеличения нагрузки есть и еще одна причина, по которой стоит использовать больше одного сервера – это повышение надежности системы.

Система J2EE, включающая библиотеки, наборы инструментальных средств и интерфейсов API, ориентирована на создание многоуровневых крупномасштабных web-приложений. Как показано на рис. 1, в подобных web-приложениях действия сервлетов Java и JSP-страниц сводятся к созданию пользовательского интерфейса, в то время как «конечными» процессами, связанными с бизнес-логикой и доступом к базам данных, управляет Enterprise JavaBean (EJB) – интерфейс API, основанный на Java.

EJB работает в специализированной среде – контейнере EJB, аналогично тому, как сервлеты работают в контейнере сервлетов. Но контейнер EJB может выполнять и многие другие функции для приложения. Функции такие, как сохранение данных сеанса и осуществление транзакций в базах данных. Теоретически контейнер EJB отвечает за все вопросы масштабирования.

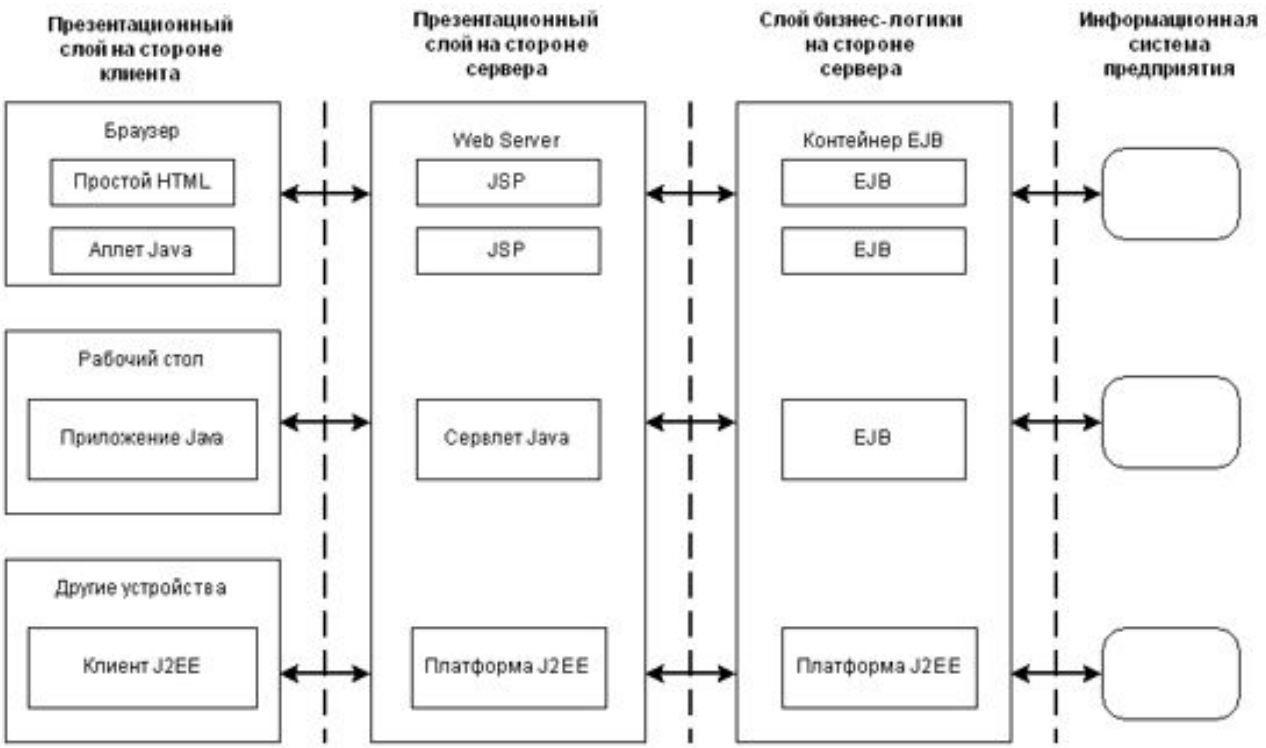

Рис. 1. Модель приложения на основе J2EE

В отличие от жестко структурированной среды J2EE в концепции Spaces web-приложение создается как слабосвязанная система. Эта концепция несколько лет обсуждалась в академических кругах, а недавно было найдено новое ее применение. Проект JINI, разработанный Sun для связи между распределенными приложениями, использует новый интерфейс API JavaSpaces.

Приложения, взаимодействующие через JavaSpaces, обмениваются сообщениями не напрямую, а посредством специального приложения, которое управляет пространством сообщений. Сообщения содержат данные, т.е. объекты Java, и идентифицирующие пометки, которые требуются, чтобы сопоставить заданию соответствующее приложение, способное это задание выполнить. Приложение, отправившее сообщение в пространство сообщений, может быть проинформировано о том, что посланный им объект прошел требуемую обработку. Сравнивая эту ситуацию с той, в которой имеется специальный диспетчер, назначающий задания приложениям, мы видим, что взаимодействие, основанное на использовании пространства сообщений, автоматически распределяет нагрузку между различными системами более равномерно. Таким образом, обращение к базе будет быстродействующим и удобным в обращении.

Список литературы

1. Фролов А.В., Фролов Г.В. Библиотека системного программиста. – М.: ДИАЛОГ-МИФИ. – Т.29.
2. Сервер Web своими руками. Язык HTML, приложения CGI и ISAPI, установка серверов Web для Windows, 1997.
3. Родли Д. Создание Java-апплетов / Пер. с англ. – К.: НИПФ “ДиаСофт Лтд.”, 1996.
4. Джамса К. Java / Пер. с англ. – Мн.: ООО “Поппури”, 1996.
5. Баженова И.Ю. Язык программирования Java. – М.: ДИАЛОГ-МИФИ, 1997.