Программирование на языке Java. Сетевые средства

Картузов А.В.

Эта глава посвящена описанию пакета java.net. Java поддерживает протокол TCP/IP, во-первых, расширяя свой интерфейс потоков ввода-вывода, описанного в предыдущей главе, и во вторых, добавляя возможности, необходимые для построения объектов ввода-вывода при работе в сети.

1. InetAddress

Java поддерживает адреса абонентов, принятые в Internet, с помощью класса InetAddress. Для адресации в Internet используются служебные функции, работающие с обычными, легко запоминающимися символическими именами, эти функции преобразуют символические имена в 32-битные адреса.

1.1. Фабричные методы

В классе InetAddress нет доступных пользователю конструкторов. Для создания объектов этого класса нужно воспользоваться одним из его фабричных методов. Фабричные методы—это обычные статические методы, которые возвращают ссылку на объект класса, которому они принадлежат. В данном случае, у класса InetAddress есть три метода, которые можно использовать для создания представителей. Это методы getLocalHost, getByName и

В приведенном ниже примере выводятся адреса и имена локальной машины, локального почтового узла и WWW-узла компании, в которой работает автор.

```
        InetAddress Address = InetAddress.getLocalHost();
        System.out.println(Address);
```

```
        Address = InetAddress.getByName("mailhost");
        System.out.println(Address);
        InetAddress SW[] = InetAddress.getAllByNarne("www.starwave.com");
        System.out.println(SW);
```

У класса InetAddress также есть несколько нестатических методов, которые можно использовать с объектами, возвращаемыми только что названными фабричными методами:

- getHostName() возвращает строку, содержащую символическое имя узла, соответствующее хранящемуся в данном объекте адресу Internet.
- getAddress() возвращает байтовый массив из четырех элементов, в котором в порядке, используемом в сети, записан адрес Internet, хранящийся в данном объекте.
- toString() возвращает строку, в которой записано имя узла и его адрес.

2. Дейтаграммы

Дейтаграммы, или пакеты протокола UDP (User Datagram Protocol)—это пакеты информации, пересылаемые в сети по принципу "fire-and-forget" (выстрелил и забыл). Если вам надо добиться оптимальной производительности, и вы в состоянии минимизировать затраты на проверку целостности

UDP не предусматривает проверок и подтверждений при передаче информации. При передаче пакета UDP по какому-либо адресу нет никакой гарантии того, что он будет принят, и даже того, что по этому адресу вообще есть кому принимать такие пакеты. Аналогично, когда вы получаете дейтаграмму, у вас нет никаких гарантий, что она не была повреждена в пути или что ее отправитель все еще ждет от вас подтверждения ее получения.

Java реализует дейтаграммы на базе протокола UDP, используя для этого два класса. Объекты класса DatagramPacket представляют собой контейнеры с данными, а DatagramSocket—это механизм, получении объектов DatagramPacket.

3. Сокеты "для клиентов"

TCP/IP-сокеты используются для реализации надежных двунаправленных, ориентированных на работу с потоками соединений точка-точка между узлами Internet.

Сокеты можно использовать для соединения системы ввода-вывода Java с программами, которые могут выполняться либо на локальной машине, либо на любом другом узле Internet. В отличие от класса DatagramSocket, объекты класса Socket реализуют высоконадежные устойчивые соединения между клиентом и сервером.

В пакете java.net классы Socket и ServerSocket сильно отличаются друг от друга. Первое отличие в том, что ServerSocket ждет, пока клиент не установит с ним соединение, в то время, как обычный Socket трактует недоступность чего-либо, с чем он хочет соединиться, как ошибку. Одновременно с созданием объекта Socket устанавливается соединение между узлами Internet. Для создания сокетов вы можете использовать два конструктора:

Socket(String host, int port) устанавливает соединение между локальной машиной и указанным портом узла Internet, имя которого было передано конструктору. Этот конструктор может возбуждать исключения UnknownHostException и IOException.

Socket(InetAddress address, int port) выполняет ту же работу, что и первый конструктор, но узел, с которым требуется установить соединение, задается не строкой, а объектом InetAddress. Этот конструктор может возбуждать только IOException.

Из объекта Socket в любое время можно извлечь информацию об адресе Internet и номере порта, с которым он соединен. Для этого служат следующие методы:

- getInetAddressQ возвращает объект InetAddress, связанный с данным объектом Socket.
- getPort() возвращает номер порта на удаленном узле, с которым установлено соединение.
- getLocalPort() возвращает номер локального порта, к которому присоединен данный объект.

После того, как объект Socket создан, им можно воспользоваться для того, чтобы получить доступ к связанным с ним входному и выходному потокам. Эти потоки используются для приема и передачи данных точно так же, как и обычные потоки ввода-вывода, которые мы видели в предыдущей главе:

- getInputStream() возвращает InputStream, связанный с данным объектом.
- getOutputStream() возвращает OutputStream, связанный с данным объектом.

- close() закрывает входной и выходной потоки объекта Socket.

Приведенный ниже очень простой пример открывает соединение с портом 880 сервера "timehost" и выводит полученные от него данные.

```
        import java.net.*;
        import java.io.*;
        class TimeHost {
                public static void main(String args[]) throws Exception {
                int c;
                Socket s = new Socket("timehost.starwave.com",880);
                InputStream in = s.getInputStream();
                while ((c = in.read()) != -1) {
                System.out.print( (char) c);
                }
        s.close();
        }
}
```

4. Сокеты "для серверов"

Как уже упоминалось ранее, Java поддерживает сокеты серверов. Для создания серверов Internet надо использовать объекты класса ServerSocket. Когда вы создаете объект ServerSocket, он регистрирует себя в системе, говоря о том, что он готов обслуживать соединения клиентов. У этого класса есть один дополнительный метод accept(), вызов которого блокирует подпроцесс до тех пор, пока какой-нибудь клиент не установит соединение по соответствующему порту. После того, как соединение установлено, метод accept() возвращает вызвавшему его подпроцессу обычный объект Socket.

Два конструктора класса ServerSocket позволяют задать, по какому порту вы хотите соединяться с клиентами, и (необязательный параметр) как долго вы готовы ждать, пока этот порт не освободится.

ServerSocket(int port) создает сокет сервера для заданного порта.

ServerSocket(int port, int count) создает сокет сервера для заданного порта. Если этот порт занят, метод будет ждать его освобождения максимум count миллисекунд.

5. URL

URL (Uniform Resource Locators—однородные указатели ресурсов)—являются наиболее фундаментальным компонентом "Всемирной паутины". Класс URL предоставляет простой и лаконичный программный интерфейс для доступа к информации в Internet с помощью URL.

У класса URL из библиотеки Java—четыре конструктора. В наиболее часто используемой форме конструктора URL адрес ресурса задается в строке, идентичной той, которую вы используете при работе с браузером:

```
URL(String spec)
```

Две следующих разновидности конструкторов позволяют задать URL, указав его отдельные компоненты:

```
URL(String protocol, String host, int port, String file)
URL(String protocol, String host, String file)
```

Четвертая, и последняя форма конструктора позволяет использовать существующий URL в качестве ссылочного контекста, и создать на основе этого контекста новый URL.

```
URL(URL context, String spec)
```

В приведенном ниже примере создается URL, адресующий www-страницу (поставьте туда свой адрес), после чего программа печатает свойства этого объекта.

```
import java.net.URL;
        class myURL {
                public static void main(String args[]) throws Exception {
                URL hp = new URL("http://coop.chuvashia.edu");
                System.out. println("Protocol: " + hp.getProtocol());
                System.out.printin("Port: " + hp.getPort());
                System.out.println("Host: " + hp.getHost());
                System.out.println("File: " + hp.getFile());
                System.out.println("Ext: " + hp.toExternaLForm());
                }
        }
```

Для того, чтобы извлечь реальную информацию, адресуемую данным URL,

необходимо на основе URL создать объект URLConnection, воспользовавшись для этого методом openConnection().

6. URLConnection

URLConnection—объект, который мы используем либо для проверки свойств удаленного ресурса, адресуемого URL, либо для получения его содержимого. В приведенном ниже примере мы создаем URLConnection с помощью метода openConnection, вызванного с объектом URL. После этого мы используем

```
import java.net.*;
import java.io.*;
class localURL {
        public static void main(String args[]) throws Exception {
                int c;
                URL hp = new URL("http", "127.0.0.1", 80, "/");
                URLConnection hpCon = hp.openConnection();
                System.out.println("Date: " + hpCon.getDate());
                System.out.println("Type: " + hpCon.getContentType());
                System.out.println("Exp: " + hpCon.getExpiration());
                System.out.println( "Last M: " + hpCon.getLastModified());
                System.out.println("Length: " + hpCon.getContentLength());
                if (hpCon.getContentLength() > 0) {
                        System.out.println("=== Content ===");
                        InputStream input = hpCon.getInputStream();
                        int i=hpCon.getContentLength();
                        while (((c = input. read()) != -1) && (—i > 0)) {
                                System.out.print((char) c);
                                }
                        input.close();
                        }
                else {
                        System.out.println("No Content Available");
                        }
                }
        }
```

Эта программа устанавливает HTTP-соединение с локальным узлом по порту 80 (у вас

на машине должен быть и установлен Web-сервер) запрашивает документ по умолчанию, обычно это—. После этого программа выводит значения заголовка, запрашивает и выводит содержимое документа.

7. Сеть и только сеть

Сетевые классы Java предоставляют ясный и простой в использовании интерфейс для работы в Internet. Фундамент, заложенный в пакете java.net—хорошая база для дальнейшего развития, которая позволит Java эволюционировать вместе с Internet.