Глава 15
СЕТЕВЫЕ ПРОГРАММЫ

Поддержка Интернет

Язык Java делает сетевое программирование простым благодаря наличию специальных средств и классов. Большинство этих классов находится в пакете **`java.net`**. Сетевые классы имеют методы для установки сетевых соединений передачи запросов и сообщений. Многопоточность позволяет обрабатывать несколько соединений. Сетевые приложения используют Internet-приложения, к которым относятся Web-браузер, e-mail, сетевые новости, передача файлов. Для создания таких приложений используются сокеты, порты, протоколы TCP/IP, UDP.

Приложения клиент/сервер используют компьютер, выполняющий специальную программу-сервер, которая обычно устанавливается на удаленном компьютере и предоставляет услуги другим программам-клиентам. Клиент – это программа, получающая услуги от сервера. Клиент устанавливает соединение с сервером и пересылает серверу запрос. Сервер осуществляет прослушивание клиентов, получает и выполняет запрос после установки соединения. Результат выполнения запроса может быть возвращен сервером клиенту. Запросы и сообщения представляют собой записи, структура которых определяется используемыми протоколами.

В стеке протоколов TCP/IP используются следующие прикладные протоколы:

HTTP – Hypertext Transfer Protocol (WWW);
NNTP – Network News Transfer Protocol (группы новостей);
SMTP – Simple Mail Transfer Protocol (посылка почты);
POP3 – Post Office Protocol (чтение почты с сервера);
FTP – File Transfer Protocol (протокол передачи файлов).

Каждый компьютер из подключенных к сети по протоколу TCP/IP имеет уникальный IP-адрес, используемый для идентификации и установки соединения. Это 32-битовое число, обычно записываемое как четыре числа, разделенные точками, каждое из которых изменяется от **`0`** до **`255`**. IP-адрес может быть временным и выделяться динамически для каждого подключения или быть постоянным, как для сервера. IP-адреса используются во внутренних сетевых системах. Обычно при подключении к Internet вместо числового IP-адреса используются символьные имена (например: `www.bsu.by`), называемые именами домена. Специальная программа DNS (Domain Name Server), располагаемая на отдельном сервере, проверяет адрес и преобразует имя домена в числовой IP-адрес. Если в качестве сервера используется этот же компьютер без сетевого подключения, в качестве IP-адреса указывается **`127.0.0.1`** или **`localhost`**. Для явной идентификации услуг к IP-адресу присоединяется номер порта через двоеточие, например

217.21.43.10:443. Здесь указан номер порта **443**. Номера портов от **1** до **1024** могут быть заняты для внутреннего использования, например, если порт явно не указан, браузер воспользуется значением по умолчанию: **20** – **FTP**-данные, **21** – **FTP**-управление, **53** – **DNS**, **80** – **HTTP**, **25** – **SMTP**, **110** – **POP3**, **119** – **NNTP**. К серверу можно подключиться с помощью различных портов. Каждый порт указывает конкретное место соединения на указанном компьютере и предоставляет определенную услугу.

Для доступа к сети Internet в браузере указывается адрес URL. Адрес URL (Universal Resource Locator) состоит из двух частей – префикса протокола (http, https, ftp и т.д.) и URI (Universal Resource Identifier). URI содержит Internet-адрес, необязательный номер порта и путь к каталогу, содержащему файл, например:

http://www.bsu.by

URI не может содержать такие специальные символы, как пробелы, табуляции, возврат каретки. Их можно задавать через шестнадцатеричные коды. Например: **%20** обозначает пробел. Другие зарезервированные символы: символ **&** – разделитель аргументов, символ **?** – следует перед аргументами запросов, символ **+** – пробел, символ **#** – ссылки внутри страницы (`имя_страницы#имя_ссылки`).

Определить IP-адрес в приложеннии можно с помощью объекта класса **InetAddress** из пакета **java.net**.

Класс **InetAddress** не имеет **public**-конструкторов. Создать объект класса можно с помощью статических методов. Метод **getLocalHost()** возвращает объект класса **InetAddress**, содержащий IP-адрес и имя компьютера, на котором выполняется программа. Метод **getByName(String host)** возвращает объект класса **InetAddress**, содержащий IP-адрес по имени компьютера, используя пространство имен DNS. IP-адрес может быть временным, различным для каждого соединения, однако он остается постоянным, если соединение установлено. Метод **getByAddress(byte[] addr)** создает объект класса **InetAddress**, содержащий имя компьютера, по IP-адресу, представленному в виде массива байт. Если компьютер имеет несколько IP, то получить их можно методом **getAllByName(String host)**, возвращающим массив объектов класса **InetAddress**. Если IP для данной машины один, то массив будет содержать один элемент. Метод **getByAddress(String host, byte[] addr)** создает объект класса **InetAddress** с заданным именем и IP-адресом, не проверяя существование такого компьютера. Все эти методы являются потенциальными генераторами исключительной ситуации **UnknownHostException**, и поэтому их вызов должен быть обработан с помощью **throws** для метода или блока **try-catch**. Проверить доступ к компьютеру в данный момент можно с помощью метода **boolean isReachable(int timeout)**, который возвращает **true**, если компьютер доступен, где **timeout** – время ожидания ответа от компьютера в миллисекундах.

Следующая программа демонстрирует при наличии Internet-соединения, как получить IP-адрес текущего компьютера и IP-адрес из имени домена с помощью сервера имен доменов (DNS), к которому обращается метод **getByName()** класса **InetAddress**.

```
/* пример # 1 : вывод IP-адреса компьютера и интернет-ресурса :
InetLogic.java*/
package chapt15;
import java.net.*;

public class InetLogic {
      public static void main(String[] args) {
            InetAddress myIP = null;
            InetAddress bsuIP = null;
            try {
             myIP = InetAddress.getLocalHost();
                  // вывод IP-адреса локального компьютера
                  System.out.println("Мой IP -> "
                        + myIP.getHostAddress());
             bsuIP = InetAddress.getByName(
                              "www.bsu.by");
                  // вывод IP-адреса БГУ www.bsu.by
                  System.out.println("BSU -> "
                        + ibaIP.getHostAddress());
            } catch (UnknownHostException e) {
                  // если не удается найти IP
                  e.printStackTrace();
            }
      }
}
```

В результате будет выведено, например:

```
Мой IP -> 172.17.16.14
BSU -> 217.21.43.10
```

Метод **getLocalHost()** класса **InetAddress** создает объект **myIP** и возвращает IP-адрес компьютера.

```
/* пример # 2 : присваивание фиктивного имени реальному ресурсу, заданному
через IP : UnCheckedHost.java */
package chapt15;
import java.io.IOException;
import java.net.InetAddress;
import java.net.UnknownHostException;

public class UnCheckedHost {
  public static void main(String[] args) {
      // задание IP-адреса лаборатории bsu.iba.by в виде массива
      byte ip[] ={(byte)217, (byte)21, (byte)43, (byte)10};
      try {
            InetAddress addr =
               InetAddress.getByAddress("University", ip);
            System.out.println(addr.getHostName()
              + "-> соединение:" + addr.isReachable(1000));
      } catch (UnknownHostException e) {
```

```
            System.out.println("адрес недоступен");
            e.printStackTrace();
      } catch (IOException e) {
            System.out.println("ошибка потока");
            e.printStackTrace();
      }
   }
}
```

В результате будет выведено в случае подключения к сети Интернет:

```
University-> соединение:true
```

Для доступа к ресурсам можно создать объект класса **URL**, указывающий на ресурсы в Internet. В следующем примере объект **URL** используется для доступа к HTML-файлу, на который он указывает, и отображает его в окне браузера с помощью метода **showDocument()**.

```
/* пример # 3 : запуск страницы из апплета: MyShowDocument.java */
package chapt15;
import java.awt.Graphics;
import java.net.MalformedURLException;
import java.net.URL;
import javax.swing.JApplet;

public class MyShowDocument extends JApplet {
      private URL bsu = null;

      public String getMyURL() {
            return "http://www.bsu.by";
      }
      public void paint(Graphics g) {
            int  timer = 0;
            g.drawString("Загрузка страницы", 10, 10);
            try {
                  for (; timer < 30; timer++) {
                    g.drawString(".", 10 + timer * 5, 25);
                    Thread.sleep(100);
                  }
                  bsu = new URL(getMyURL());
      getAppletContext().showDocument(bsu, "_blank");
            } catch (InterruptedException e) {
                  e.printStackTrace();
            } catch (MalformedURLException e) {
            // некорректно задан протокол, доменное имя или путь к файлу
                  e.printStackTrace();
            }
      }
}
```

Метод **showDocument()** может содержать параметры для отображения страницы различными способами: "**_self**" – выводит документ в текущий фрейм,

"**_blank**" – в новое окно, "**_top**" – на все окно, "**_parent**" – в родительском окне, "**имя_окна**" – в окне с указанным именем. Для корректной работы данного примера апплет следует запускать из браузера, используя следующий HTML-документ:

```
<html>
<body align=center>
<applet code=chapt15.MyShowDocument.class></applet>
</body></html>
```

В следующей программе читается содержимое HTML-файла по указанному адресу и выводится в окно консоли.

```
/* пример # 4 : чтение документа из интернета: ReadDocument.java */
package chapt15;
import java.net.*;
import java.io.*;

public class ReadDocument {
   public static void main(String[] args) {
      try {
            URL lab = new URL("http://www.bsu.by");
            InputStreamReader isr =
                  new InputStreamReader(lab.openStream());
            BufferedReader d = new BufferedReader(isr);
            String line = "";
            while ((line = d.readLine()) != null) {
                  System.out.println(line);
            }
      } catch (MalformedURLException e) {
      // некорректно заданы протокол, доменное имя или путь к файлу
            e.printStackTrace();
      } catch (IOException e) {
            e.printStackTrace();
      }
   }
}
```

Сокетные соединения по протоколу TCP/IP

Сокеты (сетевые разъёмы) – это логическое понятие, соответствующее разъёмам, к которым подключены сетевые компьютеры и через которые осуществляется двунаправленная поточная передача данных между компьютерами. Сокет определяется номером порта и IP-адресом. При этом IP-адрес используется для идентификации компьютера, номер порта – для идентификации процесса, работающего на компьютере. Когда одно приложение знает сокет другого, создается сокетное протоколо-ориентированное соединение по протоколу TCP/IP. Клиент пытается соединиться с сервером, инициализируя сокетное соединение. Сервер прослушивает сообщение и ждет, пока клиент не свяжется с ним. Первое сообщение, посылаемое клиентом на сервер, содержит сокет клиента. Сервер, в свою очередь, создает сокет, который будет использоваться для связи с клиентом, и