QST 青软实训

# 第6章 网络编程

# 本章重点



- 掌握Socket类及其方法的使用
- 掌握ServerSocket类的使用
- 了解网络基础相关的概念
- 了解Java 网络相关的API

# 任务驱动



- **本章任务是使用网络编程完成“Q-DMS数据挖掘”系统的数据发送功能：**

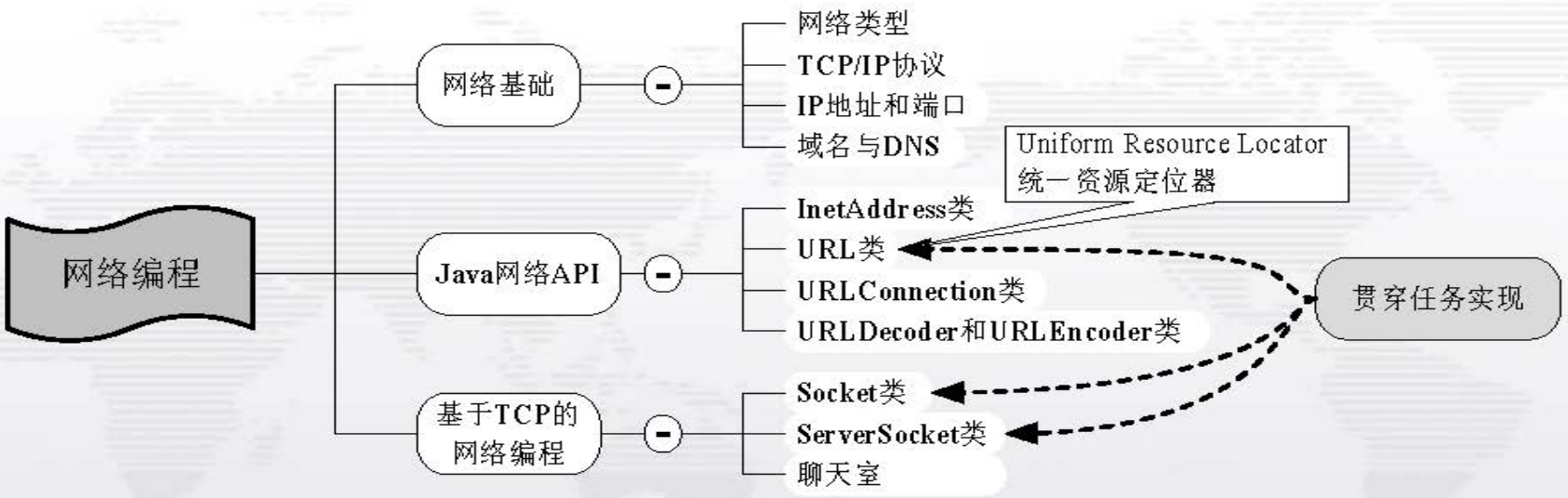
【任务6-1】 使用Socket实现主窗口中的客户端数据发送到服务器的功能。

【任务6-2】 使用ServerSocket实现服务器端应用程序，实现接收所有客户端发送的日志和物流信息，并将信息保存到数据库。

【任务6-3】 运行服务器及客户端应用程序，演示多客户端的数据发送效果。

# 学习路线




网络编程
网络基础
网络类型
TCP/IP协议
IP地址和端口
域名与DNS
Java网络API
InetAddress类
URL类
URLConnection类
URLDecoder和URLEncoder类
Uniform Resource Locator
统一资源定位器
基于TCP的
网络编程
Socket类
ServerSocket类
聊天室
贯穿任务实现

# 本章目标



| 知识点 | 听 | 看 | 抄 | 改 | 写 |
| --- | --- | --- | --- | --- | --- |
| 网络基础 | ★ | ★ | | | |
| Java网络API | ★ | ★ | ★ | ★ | |
| 基于TCP的网络编程 | ★ | ★ | ★ | ★ | ★ |

# 6.1.1 网络类型



根据规模大小、地理位置以及延伸范围对计算机网络进行分类

- 局域网（LAN）
- 城域网（MAN）
- 广域网（WAN）

## 按照网络的拓扑结构可分为

- 星型网络
- 总线型网络
- 环型网络
- 树型网络
- 分布式网络
- 网状网络
- 蜂窝状网络
- 混合型网络

# 6.1.2 TCP/IP协议



- 在计算机网络中实现通信必须遵守一些约定，即通信协议
- 通信协议规定了通信的内容、方式和通信时间，其核心要素有:
  - 语义
  - 语法
  - 时序
- 常见的通信协议包括:
  - TCP/IP协议
  - IPX/SPX协议
  - NetBEUI协议
  - RS-232-C协议、V.35等

# 6.1.3 IP地址和端口



IP地址用于唯一地标识网络中的一个通信实体，IP地址被分成五类

- A类地址：范围1.0.0.0~127.255.255.255
- B类地址：范围128.0.0.0~191.255.255.255
- C类地址：范围192.0.0.0~223.255.255.255
- D类地址：范围从224.0.0.0到239.255.255.255
- E类地址：范围从240.0.0.0到255.255.255.255

## A、B、C三类地址是由Internet NIC在全球范围内统一分配，其最大网络数及范围如表所示

| 类别 | 最大网络数 | IP地址范围 | 最大主机数 | 私有IP地址范围 |
| --- | --- | --- | --- | --- |
| A | 126（2^7-2) | 0.0.0.0~127.255.255.255 | 16777214 | 10.0.0.0~10.255.255.255 |
| B | 16384(2^14) | 128.0.0.0~191.255.255.255 | 65534 | 172.16.0.0~172.31.255.255 |
| C | 2097152(2^21) | 192.0.0.0~223.255.255.255 | 254 | 192.168.0.0~192.168.255.255 |

一些特殊的网址：

- 每一个字节都为0的地址（0.0.0.0）对应于当前主机
- 每个字节都为1的地址（255.255.255.255）是当前子网的广播地址
- 以“11110”开头的E类IP地址都保留用于将来和实验使用
- 地址中数字127.0.0.1到127.255.255.255用于回路测试
- 网络ID的第一个8位组也不能全置为“0”，全“0”表示本地网络

# 端口



端口是应用程序与外界交流的出入口，用于表示数据交给哪个通信程序进行处理，通常将端口分为三类

- 公认端口（Well Known Ports）：从0到1023
- 注册端口（Registered Ports）：从1024到49151
- 动态和/或私有端口（Dynamic and/or PrivatePorts）：从49152到65535

# 端口是通过端口号来标记的，常用端口及其对应服务

| 端口号 | 服务 |
| --- | --- |
| 7 | Echo服务端口 |
| 21 | FTP服务端口 |
| 23 | Telnet服务端口 |
| 25 | SMTP服务端口 |
| 80 | HTTP服务端口 |

**注意**

自己编写的应用程序尽量避免使用表中的公认端口值。

# 6.1.4 域名与DNS



- 域名是由一串用点分隔的字符串所组成的Internet上某一台计算机或计算机组的名称
- 域名由两个或两个以上的词构成，中间由“.”号分隔开
- 通常由特定字符集、英文字母、数字及“-”任意组合而成，但开头和结尾均不能含有“-”符号

注意

域名中的字母不分大小写，最长可达67个字节。域名不仅便于记忆，而且即使在IP地址发生变化的情况下，通过改变解析对应关系，域名仍可保持不变。目前也有一些其他语言的域名，如中文域名。

# 域名



按照级别，可以将域名分为：

- 顶级域名
  - 国际顶级域名（iTDs）
  - 国家顶级域名（nTLDs）
- 二级域名
- 三级域名

# DNS



- DNS（Domain Name System，域名系统）是域名与IP地址之间相互映射的一个分布式数据库
- 通过DNS可以将用户输入的域名解析为与之相关IP地址

# 6.2 Java网络API



- Java中有关网络方面的功能都定义在java.net包中
- URL和URLConnection等类提供了以程序的方式来访问Web服务
- URLDecoder和URLEncoder提供了字符串相互转换的静态方法

# 6.2.1 InetAddress类



- Java提供InetAddress类来封装IP地址或域名
- InetAddress类无构造方法，因此不能直接创建其对象

而是通过静态方法来创建InetAddress对象或数组

- InetAddress类常用方法

| 方法 | 功能描述 |
| --- | --- |
| public static InetAddress getLocalHost() | 获得本机对应的InetAddress对象 |
| public static InetAddress getByName (String host) | 根据主机获得对应的InetAddress对象，参数host可以是IP地址或域名 |
| public static InetAddress[] getAllByName(String host) | 根据主机获得具有相同名字的一组InetAddress对象 |
| public static InetAddress getByAddress(byte[] addr) | 获取addr所封装的IP地址对应的InetAddress对象 |
| public String getCanonicalHostName() | 获取此IP地址的全限定域名 |
| public bytes[] getHostAddress() | 获得该InetAddress对象对应的IP地址字符串 |
| public String getHostName() | 获得该InetAddress对象的主机名称 |
| public boolean isReachable(int timeout) | 判断是否可以到达该地址 |

**讲师演示讲解**

**【代码6-1】InetAddressDemo.java**

**注意**

在获得Internet上的域名所对应的地址信息时，需保证运行环境能访问Internet，否则将抛出UnknownHostException异常。

# 6.2.2 URL类



- URL（Uniform Resource Locator，统一资源定位器）表示互联网上某一资源的地址
- URL可以由协议名、主机、端口和资源四个部分组成，其语法

```
protocol://host:port/resourceName
```

  - protocol是协议名
  - host是主机名
  - port是端口
  - resourceName是资源名
- 【示例】URL地址

```
http://book.moocollege.cn/java-book1.html
```

- Java将URL封装成URL类，通过URL对象记录下完整的URL信息
- URL类常用方法

| 方法 | 功能描述 |
| --- | --- |
| public URL(String spec) | 构造方法，根据指定的字符串来创建一个URL对象 |
| Public URL(String protocol,String host,int port,String file) | 构造方法，根据指定的协议、主机名、端口号和文件资源来创建一个URL对象 |
| public URL(String protocol, String host, String file) | 构造方法，根据指定的协议、主机名、和文件资源来创建URL对象 |
| public String getProtocol() | 返回协议名 |
| public String getHost() | 返回主机名 |
| public int getPort() | 返回端口号，如果没有设置端口，则返回-1 |
| public String getFile() | 返回文件名 |
| public String getRef() | 返回URL的锚 |
| public String getQuery() | 返回URL的查询信息 |
| public String getPath() | 返回URL的路径 |
| public URLConnection openConnection() | 返回一个URLConnection对象 |
| public final InputStream openStream() | 返回一个用于读取该URL资源的InputStream流 |

注意

JDK还提供了一个URI（Uniform Resource Identifiers，统一资源标识符）类，该类的实例不能用于定位任何资源，其唯一作用就是解析，可以将URL理解成URI的特例。URL类的构造方法都声明抛出异常MalformedURLException，因此在创建URL对象时，需要对该异常进行处理，即new URL()需要放在try...catch语句中捕获该异常并处理，或者在方法后使用throws显式声明抛出该异常。

**讲师演示讲解**

**【代码6-2】URLDemo.java**

# 6.2.3 URLConnection类



- URLConnection代表与URL指定的数据源的动态连接
- 允许使用POST或PUT和其他HTTP请求方法将数据送回服务器
- URLConnection常用方法

| 方法 | 功能描述 |
| --- | --- |
| public int getContentLength() | 获得文件的长度 |
| public String getContentType() | 获得文件的类型 |
| public long getDate() | 获得文件创建的时间 |
| public long getLastModified() | 获得文件最后修改的时间 |
| public InputStream getInputStream() | 获得输入流，以便读取文件的数据 |
| public OutputStream getOutputStream() | 获得输出流，以便输出数据 |
| public void setRequestProperty(String key,String value) | 设置请求属性值 |

**讲师演示讲解**

**【代码6-3】URLConnectionDemo.java**

**注意**

Java 8新增一个URLPermission工具类，用于管理URLConnection的权限问题，如果在URLConnection安装了安全管理器，通过该对象打开连接时先需要获得权限。

# 6.2.4 URLDecoder和URLEncoder类



- 在编程过程中涉及到普通字符串和application/x-www-form-urlencoded MIME字符串之间相互转换时

这种编码：当URL地址中包含西欧字符时，系统会将这些非西欧字符转换成特殊编码如“%XX”格式

- 需要使用URLDecoder和URLEncoder两个工具类
  - URLDecoder类提供了decode(String s,String enc)静态方法
  - URLEncoder类提供了encode(String s,String enc)静态方法

s参数：表示要解码的MIME字符串
enc参数：指定编码格式

**讲师演示讲解**

**【代码6-4】URLDecoderDemo.java**

# 6.3 基于TCP的网络编程



- 使用TCP/IP协议进行通信时，会在通信的两端各建立一个Socket（套接字），从而在通信的两端之间形成网络虚拟链路


TCP/IP协议通信原理

- Java对基于TCP的网络通信提供了封装，使用Socket对象封装了两端的通信端口
- Socket允许应用程序将网络连接当成一个IO流
- java.net包中提供了网络编程所需的类，其中基于TCP协议的网络编程主要使用两种Socket
  - ServerSocket：是服务器套接字
  - Socket：是客户端套接字

# 6.3.1 Socket类



## 通常客户端使用Socket来连接指定的服务器，常用的构造方法

- Socket(InetAddress |String host,int port)
- Socket(InetAddress|String host,int port,InetAddress localAddr,int localPort)

创建连接到指定远程主机和端口号的Socket对象，默认使用本地主机Ip和系统动态分配的端口

创建连接到指定主机和端口号的Socket对象，并指定本地IP地址和本地端口号，适用于本地主机有多个IP地址的情况。

### 【示例】创建Socket对象

```
try{
  Socket s= new Socket("192.168.1.128" , 28888);
  ...//Socket通信
}catch (IOException e) {
  e.printStackTrace();
}
```

注意

上述两个Socket构造方法都声明抛出IOException异常，因此在创建Socket对象必须捕获或抛出异常。端口号建议采用注册端口（范围是1024~49151之间的数），通常应用程序使用该范围内的端口，以防止发生冲突。

# Socket类常用方法

| 方法 | 功能描述 |
| --- | --- |
| public InetAddress getInetAddress() | 返回连接到远程主机的地址，如果连接失败则返回以前连接的主机 |
| public int getPort() | 返回Socket连接到远程主机的端口号 |
| public int getLocalPort() | 返回本地连接终端的端口号 |
| public InputStream getInputStream() | 返回一个输入流，从Socket读取数据 |
| public OutputStream getOutputStream() | 返回一个输出流，往Socket中写数据 |
| public synchronized void close() | 关闭当前Socket连接 |

## 使用Socket进行网络通信的具体步骤：

① 根据指定IP地址和端口号创建一个Socket对象

② 调用getInputStream()方法或getOutputStream()方法打开连接到Socket的输入/出流

③ 客户端与服务器根据协议进行交互，直到关闭连接

④ 关闭客户端的Socket

讲师演示讲解

【代码6-5】ClientSocketDemo.java

# 6.3.2 ServerSocket类



- ServerSocket是服务器套接字，运行在服务器端，通过指定端口主动监听来自客户端的Socket连接
- ServerSocket类常用的构造方法：
  - ServerSocket(int port)
  - ServerSocket(int port,int backlog)
  - ServerSocket(int port,int backlog,InetAddress localAddr)

注意

ServerSocket类的构造方法都声明抛出IOException异常，因此在创建ServerSocket对象必须捕获或抛出异常。另外，在选择端口号时，建议选择注册端口（范围是1024~49151的数），通常应用程序使用这个范围内的端口，以防止发生冲突。

## 【示例】创建ServerSocket对象

```
try {
        ServerSocket server = new ServerSocket(28888);
} catch (IOException e) {
        e.printStackTrace();
}
```

## ServerSocket常用的方法

| 方法名 | 功能说明 |
| --- | --- |
| public Socket accept() | 接收客户端Socket连接请求，并返回一个与客户端Socket对应的Socket实例；该方法是一个阻塞方法，如果没有接收到客户端发送的Socket，则一直处于等待状态，线程也会被阻塞 |
| public InetAddress getInetAddress() | 返回当前ServerSocket实例的地址信息 |
| public int getLocalPort() | 返回当前ServerSocket实例的服务端口 |
| public void close() | 关闭当前ServerSocket实例 |

使用ServerSocket进行网络通信的具体步骤：

① 根据指定的端口号来实例化一个ServerSocket对象

② 调用ServerSocket对象的accept()方法接收客户端发送的Socket对象

③ 调用Socket对象的getInputStream()/getOutputStream()方法来建立与客户端进行交互的IO流

④ 服务器与客户端根据一定的协议交互，直到关闭连接

⑤ 关闭服务器端的Socket

⑥ 回到第2步，继续监听下一次客户端发送的Socket请求连接

**讲师演示讲解**

**【代码6-6】ServerSocketDemo.java**

**注意**

在局域网环境下，可以选择其中的一台计算机作为服务器，运行服务器端ServerSocketDemo应用程序。在局域网中另外一台计算机上修改并运行客户端ClientSocketDemo应用程序，将Socket的IP地址改为服务器的IP地址；当程序运行时不同的客户端将发送不同的用户名给服务器，如此可以观察到基于C/S架构的网络通信。

使用Socket进行基于C/S架构的网络通信程序设计的过程：

① 服务器端通过某个端口监听是否有客户端发送Socket连接请求

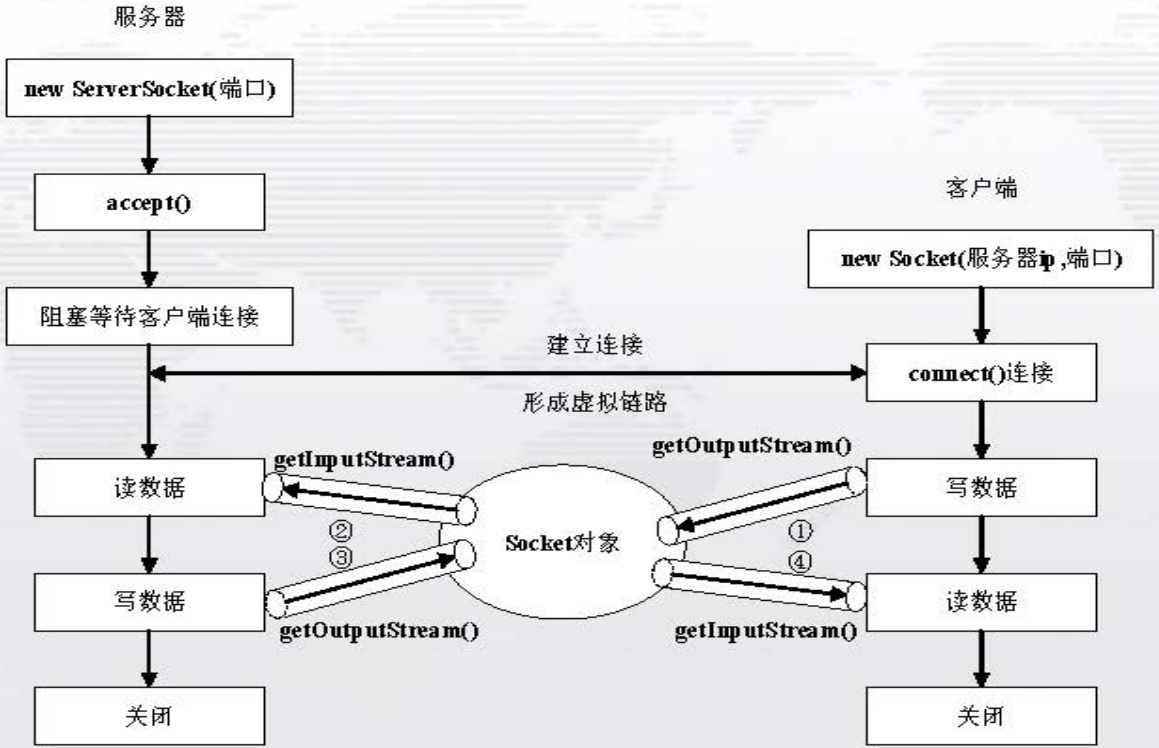
② 客户端向服务器端发出一个Socket连接请求

③ 服务器端调用accept()接收客户端Socket并建立连接

④ 通过调用Socket对象的getInputStream()/getOutStream()方法进行IO流操作，服务器与客户端之间进行信息交互

⑤ 关闭服务器端和客户端的Socket

# 服务器和客户端使用Socket交互编程模型：



## 实现客户端与服务器之间的信息交互的步骤：

① 客户端调用Socket对象的getOutputStream()方法获取输出流

② 服务器端调用Socket对象的getInputStream()方法获取输入流

③ 服务器端调用Socket对象的getOutputStream()方法获取输出流

④ 最后客户端调用Socket对象的getInputStream()方法获取输入流

**注意**

获取套接字的输入流和输出流，都是站在内存立场上考虑的，而不是套接字的立场。例如，getInputStream()方法获取套接字的输入流，用于读取Socket数据，并将数据存入到内存中。

# 6.3.3 聊天室



- 使用Socket和ServerSocket实现多人聊天的聊天室程序
- 聊天室程序是基于C/S架构，分客户端代码和服务器端代码

讲师演示讲解

【代码6-7】ChatClient.java 【代码6-8】ChatServer.java

注意

在局域网环境中，需要指定其中的一台计算机作为服务器并运行服务器端应用程序；修改客户端程序，将创建Socket的本机IP“127.0.0.1”改为服务器的真正IP地址，然后在其他不同的计算机上运行客户端应用程序，可以更好地测试该聊天室应用程序。

# 6.4 贯穿任务实现



- 【任务6-1】使用Socket实现主窗口中的客户端数据发送到服务器的功能。
  - *oracle.properties*
  - *MainFrame.java*
- 【任务6-2】使用ServerSocket实现服务器端应用程序，实现接收所有客户端发送的日志和物流信息，并将信息保存到数据库。
  - *DmsNetServer.java*

# 本章总结



- TCP/IP协议提供一种数据打包和寻址的标准方法，可以在Internet中无差错地传送数据
- 域名是由一串用点分隔的字符串组成的Internet上某一台计算机或计算机组的名称
- DNS（Domain Name Server）是进行域名解析的服务器
- URL(Uniform Resource Locator)是统一资源定位器的简称
- URL的组成：协议名://机器名：端口号/文件名/内部引用
- URLConnection是一个抽象类，代表与URL指定的数据源的动态连接
- 网络上的两个程序通过Socket实现双向通讯和数据交换

# 本章总结



- Socket和ServerSocket分别用来表示双向连接的客户端和服务端
- 在创建Socket或ServerSocket时必须捕获或声明异常
- 在Socket对象使用完毕时，要将其关闭，并且遵循一定的关闭次序

QST 青软实训

# 信·未来