---

package java.net

Interface Index

- ContentHandlerFactory
- SocketImplFactory
- URLStreamHandlerFactory

Class Index

- ContentHandler
- DatagramPacket
- DatagramSocket
- InetAddress
- ServerSocket
- Socket
- SocketImpl
- URL
- URLConnection
- URLEncoder
- URLStreamHandler

Exception Index

- MalformedURLException
- ProtocolException
- SocketException
- UnknownHostException
- UnknownServiceException

Class java.net.ContentHandler

```
java.lang.Object
   |
   +----java.net.ContentHandler
```

---

public class **ContentHandler**
extends Object

A class to read data from a URLConnection and construct an Object. Specific subclasses of ContentHandler handle specific mime types. It is the responsibility of a ContentHandlerFactor to select an appropriate ContentHandler for the mime–type of the URLConnection. Applications should never call ContentHandlers directly, rather they should use URL.getContent() or URLConnection.getContent()

Constructor Index

- **ContentHandler**()

Method Index

- **getContent**(URLConnection)
  Given an input stream positioned at the beginning of the representation of an object, read that stream and recreate the object from it

Constructors

ContentHandler

```
public ContentHandler()
```

Methods

getContent

```
public abstract Object getContent(URLConnection urlc) throws IOException
```

Given an input stream positioned at the beginning of the representation of an object, read that stream and recreate the object from it

Throws:IOException

An IO error occurred while reading the object.

Interface java.net.ContentHandlerFactory

public interface **ContentHandlerFactory**
extends Object

This interface defines a factory for ContentHandler instances. It is used by the URLStreamHandler class to create ContentHandlers for various streams.

Method Index

- **createContentHandler**(String)
  Creates a new ContentHandler to read an object from a URLStreamHandler.

Methods

createContentHandler

```
public abstract ContentHandler createContentHandler(String mimetype)
```

Creates a new ContentHandler to read an object from a URLStreamHandler.
Parameters:
mimetype – The mime type for which a content handler is desired.

Class java.net.DatagramPacket

```
java.lang.Object
   |
   +----java.net.DatagramPacket
```

---

public final class **DatagramPacket**
extends Object

A class that represents a datagram packet containing packet data, packet length, internet addresses and port.

Constructor Index

- **DatagramPacket**(byte[], int)
  This constructor is used to create a DatagramPacket object used for receiving datagrams.
- **DatagramPacket**(byte[], int, InetAddress, int)
  This constructor is used construct the DatagramPacket to be sent.

Method Index

- **getAddress**()
- **getData**()
- **getLength**()
- **getPort**()

Constructors

DatagramPacket

```
public DatagramPacket(byte ibuf[],
                      int ilength)
```

This constructor is used to create a DatagramPacket object used for receiving datagrams.
Parameters:

ibuf – is where packet data is to be received.
ilength – is the number of bytes to be received.

DatagramPacket

```
public DatagramPacket(byte ibuf[],
                      int ilength,
                      InetAddress iaddr,
                      int iport)
```

This constructor is used construct the DatagramPacket to be sent.
Parameters:
ibuf – contains the packet data.
ilength – contains the packet length
iaddr – and iport contains destination ip addr and port number.

Methods

getAddress

```
public InetAddress getAddress()
```

getPort

```
public int getPort()
```

getData

```
public byte[] getData()
```

getLength

```
public int getLength()
```

Class java.net.DatagramSocket

```
java.lang.Object
   |
   +----java.net.DatagramSocket
```

---

public class **DatagramSocket**
extends Object

The datagram socket class implements unreliable datagrams.

Constructor Index

- **DatagramSocket**()
  Creates a datagram socket
- **DatagramSocket**(int)
  Creates a datagram socket

Method Index

- **close**()
  Close the datagram socket.
- **finalize**()
  Code to perform when this object is garbage collected.
- **getLocalPort**()
  Returns the local port that this socket is bound to.
- **receive**(DatagramPacket)
  Receives datagram packet.
- **send**(DatagramPacket)
  Sends Datagram Packet to the destination address

Constructors

DatagramSocket

```
public DatagramSocket() throws SocketException
```

Creates a datagram socket

DatagramSocket

```
public DatagramSocket(int port) throws SocketException
```

Creates a datagram socket
Parameters:
local – port to use

Methods

send

```
public void send(DatagramPacket p) throws IOException
```

Sends Datagram Packet to the destination address
Parameters:
DatagramPacket – to be sent. The packet contains the buffer of bytes, length and destination InetAddress and port.
Throws:IOException
i/o error occurred

receive

```
public void receive(DatagramPacket p) throws IOException
```

Receives datagram packet.
Parameters:
DatagramPacket – to be received. On return, the DatagramPacket contains the buffer in which the data is received, packet length, sender's address and sender's port number. Blocks until some input is available.
Throws:IOException
i/o error occurred

getLocalPort

```
public int getLocalPort()
```

Returns the local port that this socket is bound to.

close

```
public synchronized void close()
```

Close the datagram socket.

finalize

```
protected synchronized void finalize()
```

Code to perform when this object is garbage collected.

Overrides:

finalize in class Object

Class java.net.InetAddress

```
java.lang.Object
   |
   +----java.net.InetAddress
```

---

public final class **InetAddress**
extends Object

A class that represents internet addresses.

Method Index

- **equals**(Object)
  Compares this object against the specified object.
- **getAddress**()
  Returns the raw IP address in network byte order.
- **getAllByName**(String)
  Given a hostname, return an array of all the corresponding InetAddresses.
- **getByName**(String)
  Returns a network address for the indicated host.
- **getHostName**()
  Gets the hostname for this address; also the key in the above hash table.
- **getLocalHost**()
  Returns the local host.
- **hashCode**()
  Returns a hashcode for this InetAddress.
- **toString**()
  Converts the InetAddress to a String.

Methods

getHostName

```
public String getHostName()
```

Gets the hostname for this address; also the key in the above hash table. If the host is equal to null, then this address refers to any of the local machine's

available network addresses.

getAddress

```
public byte[] getAddress()
```

Returns the raw IP address in network byte order. The highest order byte position is in addr[0]. An array of bytes is returned so we are prepared for 64–bit IP addresses.

Returns:

raw IP address in network byte order.

hashCode

```
public int hashCode()
```

Returns a hashcode for this InetAddress.

Overrides:

hashCode in class Object

equals

```
public boolean equals(Object obj)
```

Compares this object against the specified object.

Parameters:

obj – the object to compare with

Returns:

true if the objects are the same; false otherwise.

Overrides:

equals in class Object

toString

```
public String toString()
```

Converts the InetAddress to a String.

Overrides:

toString in class Object

getByName

```
public static synchronized InetAddress getByName(String host) throws UnknownHostException
```

Returns a network address for the indicated host. A host name of null refers to default address for the local machine. A local cache is used to speed access to addresses. If a all addresses for host are needed, use the getAllByName() method.

Parameters:

host – the specified host

Throws: UnknownHostException
If the address is unknown.

getAllByName

```
public static synchronized InetAddress[] getAllByName(String host) throws UnknownHostException
```

Given a hostname, return an array of all the corresponding InetAddresses.
Throws: UnknownHostException
If the host name could not be resolved

getLocalHost

```
public static InetAddress getLocalHost() throws UnknownHostException
```

Returns the local host.
Throws: UnknownHostException
If the host name could not be resolved

Class java.net.MalformedURLException

```
java.lang.Object
   |
   +----java.lang.Throwable
           |
           +----java.lang.Exception
                   |
                   +----java.io.IOException
                           |
                           +----java.net.MalformedURLException
```

---

public class **MalformedURLException**
extends IOException

Signals that a malformed URL has occurred.

Constructor Index

- **MalformedURLException**()
  Constructs a MalformedURLException with no detail message.
- **MalformedURLException**(String)
  Constructs a MalformedURLException with the specified detail message.

Constructors

MalformedURLException

```
public MalformedURLException()
```

Constructs a MalformedURLException with no detail message. A detail message is a String that describes this particular exception.

MalformedURLException

```
public MalformedURLException(String msg)
```

Constructs a MalformedURLException with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

msg – the detail message

Class java.net.ProtocolException

```
java.lang.Object
   |
   +----java.lang.Throwable
           |
           +----java.lang.Exception
                   |
                   +----java.io.IOException
                           |
                           +----java.net.ProtocolException
```

---

public class **ProtocolException**
extends IOException

Signals when connect gets an EPROTO. This exception is specifically caught in class Socket.

Constructor Index

- **ProtocolException**(String)
  Constructs a new ProtocolException with the specified detail message.
- **ProtocolException**()
  Constructs a new ProtocolException with no detail message.

Constructors

ProtocolException

```
public ProtocolException(String host)
```

Constructs a new ProtocolException with the specified detail message. A detail message is a String that gives a specific description of this error.

Parameters:

host – the detail message

ProtocolException

```
public ProtocolException()
```

Constructs a new ProtocolException with no detail message. A detail message is a String that gives a specific description of this error.

Class java.net.ServerSocket

```
java.lang.Object
   |
   +----java.net.ServerSocket
```

---

public final class **ServerSocket**
extends Object

The server Socket class. It uses a SocketImpl to implement the actual socket operations. It is done this way so that you are able to change socket implementations depending on the kind of firewall that is used. You can change socket implementations by setting the SocketImplFactory.

Constructor Index

- **ServerSocket**(int)
  Creates a server socket on a specified port.
- **ServerSocket**(int, int)
  Creates a server socket, binds it to the specified local port and listens to it.

Method Index

- **accept**()
  Accepts a connection.
- **close**()
  Closes the server socket.
- **getInetAddress**()
  Gets the address to which the socket is connected.
- **getLocalPort**()
  Gets the port to which the socket is listening on
- **setSocketFactory**(SocketImplFactory)
  Sets the system's server SocketImplFactory.
- **toString**()
  Returns the implementation address and implementation port of this ServerSocket as a String.

Constructors

ServerSocket

```
public ServerSocket(int port) throws IOException
```

Creates a server socket on a specified port.
Parameters:
port – the port
Throws:IOException
IO error when opening the socket.

ServerSocket

```
public ServerSocket(int port,
                    int count) throws IOException
```

Creates a server socket, binds it to the specified local port and listens to it. You can connect to an annonymous port by specifying the port number to be 0.
Parameters:
port – the specified port
count – the amt of time to listen for a connection

Methods

getInetAddress

```
public InetAddress getInetAddress()
```

Gets the address to which the socket is connected.

getLocalPort

```
public int getLocalPort()
```

Gets the port to which the socket is listening on

accept

```
public Socket accept() throws IOException
```

Accepts a connection. This method will block until the connection is made.
Throws:IOException
IO error when waiting for the connection.

close

```
public void close() throws IOException
```

Closes the server socket.
Throws:IOException
IO error when closing the socket.

toString

```
public String toString()
```

Returns the implementation address and implementation port of this ServerSocket as a String.
Overrides:
toString in class Object

setSocketFactory

```
public static synchronized void setSocketFactory(SocketImplFactory fac) throws IOException
```

Sets the system's server SocketImplFactory. The factory can be specified only once.
Parameters:
fac – the desired factory
Throws: SocketException
If the factory has already been defined.
Throws:IOException
IO error when setting the socket factor.

Class java.net.Socket

```
java.lang.Object
   |
   +----java.net.Socket
```

---

public final class **Socket**
extends Object

The client Socket class. It uses a SocketImpl to implement the actual socket operations. It is done this way so that you are able to change socket implementations depending on the kind of firewall that is used. You can change socket implementations by setting the SocketImplFactory.

Constructor Index

- **Socket**(String, int)
  Creates a stream socket and connects it to the specified port on the specified host.
- **Socket**(String, int, boolean)
  Creates a socket and connects it to the specified port on the specified host.
- **Socket**(InetAddress, int)
  Creates a stream socket and connects it to the specified address on the specified port.
- **Socket**(InetAddress, int, boolean)
  Creates a socket and connects it to the specified address on the specified port.

Method Index

- **close**()
  Closes the socket.
- **getInetAddress**()
  Gets the address to which the socket is connected.
- **getInputStream**()
  Gets an InputStream for this socket.
- **getLocalPort**()
  Gets the local port to which the socket is connected.
- **getOutputStream**()
  Gets an OutputStream for this socket.

- **getPort**()
  Gets the remote port to which the socket is connected.
- **setSocketImplFactory**(SocketImplFactory)
  Sets the system's client SocketImplFactory.
- **toString**()
  Converts the Socket to a String.

Constructors

Socket

```
public Socket(String host,
              int port) throws UnknownHostException, IOException
```

Creates a stream socket and connects it to the specified port on the specified host.

Parameters:

host – the host
port – the port

Socket

```
public Socket(String host,
              int port,
              boolean stream) throws IOException
```

Creates a socket and connects it to the specified port on the specified host. The last argument lets you specify whether you want a stream or datagram socket.

Parameters:

host – the specified host
port – the specified port
stream – a boolean indicating whether this is a stream or datagram socket

Socket

```
public Socket(InetAddress address,
              int port) throws IOException
```

Creates a stream socket and connects it to the specified address on the specified port.

Parameters:

address – the specified address
port – the specified port

Socket

```
public Socket(InetAddress address,
              int port,
              boolean stream) throws IOException
```

Creates a socket and connects it to the specified address on the specified port. The last argument lets you specify whether you want a stream or datagram socket.

Parameters:

address – the specified address
port – the specified port
stream – a boolean indicating whether this is a stream or datagram socket

- **getInetAddress**

```
public InetAddress getInetAddress()
```

Gets the address to which the socket is connected.

- **getPort**

```
public int getPort()
```

Gets the remote port to which the socket is connected.

- **getLocalPort**

```
public int getLocalPort()
```

Gets the local port to which the socket is connected.

- **getInputStream**

```
public InputStream getInputStream() throws IOException
```

Gets an InputStream for this socket.

- **getOutputStream**

```
public OutputStream getOutputStream() throws IOException
```

Gets an OutputStream for this socket.

- **close**

```
public synchronized void close() throws IOException
```

Closes the socket.

- **toString**

```
public String toString()
```

Converts the Socket to a String.

Overrides:

toString in class Object

setSocketImplFactory

```
public static synchronized void setSocketImplFactory(SocketImplFactory fac) throws IOException
```

Sets the system's client SocketImplFactory. The factory can be specified only once.

Parameters:

fac – the desired factory

Throws: SocketException

If the factory is already defined.

Class java.net.SocketException

```
java.lang.Object
   |
   +----java.lang.Throwable
           |
           +----java.lang.Exception
                   |
                   +----java.io.IOException
                           |
                           +----java.net.SocketException
```

---

public class **SocketException**
extends IOException

Signals that an error occurred while attempting to use a socket.

Constructor Index

- **SocketException**(String)
  Constructs a new SocketException with the specified detail message.
- **SocketException**()
  Constructs a new SocketException with no detail message.

Constructors

SocketException

```
public SocketException(String msg)
```

Constructs a new SocketException with the specified detail message. A detail message is a String that gives a specific description of this error.

Parameters:

msg – the detail message

SocketException

```
public SocketException()
```

Constructs a new SocketException with no detail message. A detail message is a String that gives a specific description of this error.

Class java.net.SocketImpl

```
java.lang.Object
   |
   +----java.net.SocketImpl
```

---

public class **SocketImpl**
extends Object

This is the Socket implementation class. It is an abstract class that must be subclassed to provide an actual implementation.

Variable Index

- **address**
  The internet address where the socket will make connection.
- **fd**
  The file descriptor object
- **localport**
- **port**
  The port where the socket will make connection.

Constructor Index

- **SocketImpl**()

Method Index

- **accept**(SocketImpl)
  Accepts a connection.
- **available**()
  Returns the number of bytes that can be read without blocking.
- **bind**(InetAddress, int)
  Binds the socket to the specified port on the specified host.
- **close**()
  Closes the socket.

- **connect**(String, int)
  Connects the socket to the specified port on the specified host.
- **connect**(InetAddress, int)
  Connects the socket to the specified address on the specified port.
- **create**(boolean)
  Creates a socket with a boolean that specifies whether this is a stream socket or a datagram socket.
- **getFileDescriptor**()
- **getInetAddress**()
- **getInputStream**()
  Gets an InputStream for this socket.
- **getLocalPort**()
- **getOutputStream**()
  Gets an OutputStream for this socket.
- **getPort**()
- **listen**(int)
  Listens for connections over a specified amount of time.
- **toString**()
  Returns the address and port of this Socket as a String.

fd

```
protected FileDescriptor fd
```

The file descriptor object

address

```
protected InetAddress address
```

The internet address where the socket will make connection.

port

```
protected int port
```

The port where the socket will make connection.

localport

```
protected int localport
```

Constructors

SocketImpl

```
public SocketImpl()
```

Methods

create

```
protected abstract void create(boolean stream) throws IOException
```

Creates a socket with a boolean that specifies whether this is a stream socket or a datagram socket.

Parameters:

stream – a boolean indicating whether this is a stream or datagram socket

connect

```
protected abstract void connect(String host,
                                int port) throws IOException
```

Connects the socket to the specified port on the specified host.

Parameters:

host – the specified host of the connection
port – the port where the connection is made

connect

```
protected abstract void connect(InetAddress address,
                                int port) throws IOException
```

Connects the socket to the specified address on the specified port.

Parameters:

address – the specified address of the connection
port – the specified port where connection is made

bind

```
protected abstract void bind(InetAddress host,
                             int port) throws IOException
```

Binds the socket to the specified port on the specified host.

Parameters:

host – the host
port – the port

listen

```
protected abstract void listen(int count) throws IOException
```

Listens for connections over a specified amount of time.
Parameters:
count – the amount of time this socket will listen for connections

accept

```
protected abstract void accept(SocketImpl s) throws IOException
```

Accepts a connection.
Parameters:
s – the accepted connection

getInputStream

```
protected abstract InputStream getInputStream() throws IOException
```

Gets an InputStream for this socket.

getOutputStream

```
protected abstract OutputStream getOutputStream() throws IOException
```

Gets an OutputStream for this socket.

available

```
protected abstract int available() throws IOException
```

Returns the number of bytes that can be read without blocking.

close

```
protected abstract void close() throws IOException
```

Closes the socket.

getFileDescriptor

```
protected FileDescriptor getFileDescriptor()
```

getInetAddress

```
protected InetAddress getInetAddress()
```

getPort

```
protected int getPort()
```

getLocalPort

```
protected int getLocalPort()
```

toString

```
public String toString()
```

Returns the address and port of this Socket as a String.

Overrides:

toString in class Object

Interface java.net.SocketImplFactory

public interface **SocketImplFactory**
extends Object

This interface defines a factory for SocketImpl instances. It is used by the socket class to create socket implementations that implement various policies.

Method Index

Methods

createSocketImpl

```
public abstract SocketImpl createSocketImpl()
```

Creates a new SocketImpl instance.

Class java.net.URL

```
java.lang.Object
   |
   +----java.net.URL
```

---

public final class **URL**
extends Object

Class URL represents a Uniform Reference Locator -- a reference to an object on the World Wide Web. This is a constant object, once it is created its fields cannot be changed.

Constructor Index

- **URL**(String, String, int, String)
  Creates an absolute URL from the specified protocol, host, port and file.
- **URL**(String, String, String)
  Creates an absolute URL from the specified protocol, host, and file.
- **URL**(String)
  Creates a URL from the unparsed absolute URL.
- **URL**(URL, String)
  Creates a URL from the unparsed URL in the context of the specified context.

Method Index

- **equals**(Object)
  Compares two URLs.
- **getContent**()
  Gets the contents from this opened connection.
- **getFile**()
  Gets the file name.
- **getHost**()
  Gets the host name.
- **getPort**()
  Gets the port number.
- **getProtocol**()
  Gets the protocol name.
- **getRef**()

Gets the ref.

- **hashCode**()
  Creates an integer suitable for hash table indexing.
- **openConnection**()
  Creates (if not already in existance) a URLConnection object that contains a connection to the remote object referred to by the URL.
- **openStream**()
  Opens an input stream.
- **sameFile**(URL)
  Compares two URLs, excluding the "ref" fields: sameFile is true if the true references the same remote object, but not necessarily the same subpiece of that object.
- **set**(String, String, int, String, String)
  Sets the fields of the URL.
- **setURLStreamHandlerFactory**(URLStreamHandlerFactory)
  Sets the URLStreamHandler factory.
- **toExternalForm**()
  Reverses the parsing of the URL.
- **toString**()
  Converts to a human–readable form.

Constructors

URL

```
public URL(String protocol,
           String host,
           int port,
           String file) throws MalformedURLException
```

Creates an absolute URL from the specified protocol, host, port and file.

Parameters:

protocol – the protocol to use
host – the host to connect to
port – the port at that host to connect to
file – the file on that host

Throws: MalformedURLException
If an unknown protocol is found.

URL

```
public URL(String protocol,
           String host,
           String file) throws MalformedURLException
```

Creates an absolute URL from the specified protocol, host, and file. The port number used will be the default for the protocol.

Parameters:

protocol – the protocol to use
host – the host to connect to
file – the file on that host

Throws: MalformedURLException
If an unknown protocol is found.

URL

```
public URL(String spec) throws MalformedURLException
```

Creates a URL from the unparsed absolute URL.

Parameters:
spec – the URL String to parse

URL

```
public URL(URL context,
           String spec) throws MalformedURLException
```

Creates a URL from the unparsed URL in the context of the specified context. If spec is an absolute URL, cool, otherwise, parse it in terms of the context. Context may be null (indicating no context).

Parameters:
context – the context to parse the URL to.
spec – the URL String to parse

Throws: MalformedURLException
If the protocol is equal to null.

Methods

set

```
protected void set(String protocol,
                   String host,
                   int port,
                   String file,
                   String ref)
```

Sets the fields of the URL. This is not a public method so that only URLStreamHandlers can modify URL fields. URLs are otherwise constant. REMIND: this method will be moved to URLStreamHandler

Parameters:
protocol – the protocol to use
host – the host name to connecto to
port – the protocol port to connect to
file – the specified file name on that host
ref – the reference

getPort

```
public int getPort()
```

Gets the port number. Returns –1 if the port is not set.

getProtocol

```
public String getProtocol()
```

Gets the protocol name.

getHost

```
public String getHost()
```

Gets the host name.

getFile

```
public String getFile()
```

Gets the file name.

getRef

```
public String getRef()
```

Gets the ref.

equals

```
public boolean equals(Object obj)
```

Compares two URLs.

Parameters:

obj – the URL to compare against.

Returns:

true if and only if they are equal, false otherwise.

Overrides:

equals in class Object

hashCode

```
public int hashCode()
```

Creates an integer suitable for hash table indexing.

Overrides:

hashCode in class Object

sameFile

```
public boolean sameFile(URL other)
```

Compares two URLs, excluding the "ref" fields: sameFile is true if the true references the same remote object, but not necessarily the same subpiece of that object.

Parameters:

other – the URL to compare against.

Returns:

true if and only if they are equal, false otherwise.

toString

```
public String toString()
```

Converts to a human–readable form.

Returns:

the textual representation.

Overrides:

toString in class Object

toExternalForm

```
public String toExternalForm()
```

Reverses the parsing of the URL.

Returns:

the textual representation of the fully qualified URL (ie. after the context and canonicalization have been applied).

openConnection

```
public URLConnection openConnection() throws IOException
```

Creates (if not already in existance) a URLConnection object that contains a connection to the remote object referred to by the URL. Invokes the appropriate protocol handler. Failure is indicated by throwing an exception.

Throws:IOException

If an I/O exception has occurred.

See Also:

URLConnection

openStream

```
public final InputStream openStream() throws IOException
```

Opens an input stream.

Throws:IOException

If an I/O exception has occurred.

getContent

```
public final Object getContent() throws IOException
```

Gets the contents from this opened connection.
Throws:IOException
If an I/O exception has occurred.

setURLStreamHandlerFactory

```
public static synchronized void setURLStreamHandlerFactory(URLStreamHandlerFactory fac)
```

Sets the URLStreamHandler factory.
Parameters:
fac – the desired factory
Throws: Error
If the factory has already been defined.

Class java.net.URLConnection

```
java.lang.Object
   |
   +----java.net.URLConnection
```

---

public class **URLConnection**
extends Object

A class to represent an active connection to an object represented by a URL. It is an abstract class that must be subclassed to provide implementation of connect.

Variable Index

- **allowUserInteraction**
- **connected**
- **doInput**
- **doOutput**
- **ifModifiedSince**
- **url**
- **useCaches**

Constructor Index

- **URLConnection**(URL)
  Constructs a URL connection to the specified URL.

Method Index

- **connect**()
  URLConnection objects go through two phases: first they are created, then they are connected.
- **getAllowUserInteraction**()
- **getContent**()
  Gets the object referred to by this URL.
- **getContentEncoding**()

Gets the content encoding.

- **getContentLength**()
  Gets the content length.
- **getContentType**()
  Gets the content type.
- **getDate**()
  Gets the sending date of the object.
- **getDefaultAllowUserInteraction**()
- **getDefaultRequestProperty**(String)
- **getDefaultUseCaches**()
  Set/get the default value of the UseCaches flag.
- **getDoInput**()
- **getDoOutput**()
- **getExpiration**()
  Gets the expriation date of the object.
- **getHeaderField**(String)
  Gets a header field by name.
- **getHeaderField**(int)
  Return the value for the nth header field.
- **getHeaderFieldDate**(String, long)
  Gets a header field by name.
- **getHeaderFieldInt**(String, int)
  Gets a header field by name.
- **getHeaderFieldKey**(int)
  Return the key for the nth header field.
- **getIfModifiedSince**()
- **getInputStream**()
  Calls this routine to get an InputStream that reads from the object.
- **getLastModified**()
  Gets the last modified date of the object.
- **getOutputStream**()
  Calls this routine to get an OutputStream that writes to the object.
- **getRequestProperty**(String)
- **getURL**()
  Gets the URL for this connection.
- **getUseCaches**()
- **guessContentTypeFromName**(String)
  A useful utility routine that tries to guess the content−type of an object based upon its extension.
- **guessContentTypeFromStream**(InputStream)
  This disgusting hack is used to check for files have some type that can be determined by inspection.
- **setAllowUserInteraction**(boolean)
  Some URL connections occasionally need to to interactions with the user.
- **setContentHandlerFactory**(ContentHandlerFactory)
  Sets the ContentHandler factory.
- **setDefaultAllowUserInteraction**(boolean)
  Set/get the default value of the allowUserInteraction flag.
- **setDefaultRequestProperty**(String, String)

Set/get the default value of a general request property.

- **setDefaultUseCaches**(boolean)
- **setDoInput**(boolean)
  A URL connection can be used for input and/or output.
- **setDoOutput**(boolean)
  A URL connection can be used for input and/or output.
- **setIfModifiedSince**(long)
  Some protocols support skipping fetching unless the object is newer than some time.
- **setRequestProperty**(String, String)
  Set/get a general request property.
- **setUseCaches**(boolean)
  Some protocols do caching of documents.
- **toString**()
  Returns the String representation of the URL connection.

Variables

url

```
protected URL url
```

doInput

```
protected boolean doInput
```

doOutput

```
protected boolean doOutput
```

allowUserInteraction

```
protected boolean allowUserInteraction
```

useCaches

```
protected boolean useCaches
```

ifModifiedSince

```
protected long ifModifiedSince
```

connected

```
protected boolean connected
```

Constructors

URLConnection

```
protected URLConnection(URL url)
```

Constructs a URL connection to the specified URL.
Parameters:
url – the specified URL

Methods

connect

```
public abstract void connect() throws IOException
```

URLConnection objects go through two phases: first they are created, then they are connected. After being created, and before being connected, various option can be specified (eg. doInput, UseCaches, ...). After connecting, it is an Error to try to set them. Operations that depend on being connected, like getContentLength, will implicitly perform the connection if necessary. Connecting when already connected does nothing.

getURL

```
public URL getURL()
```

Gets the URL for this connection.

getContentLength

```
public int getContentLength()
```

Gets the content length. Returns –1 if not known.

getContentType

```
public String getContentType()
```

Gets the content type. Returns null if not known.

getContentEncoding

```
public String getContentEncoding()
```

Gets the content encoding. Returns null if not known.

getExpiration

```
public long getExpiration()
```

Gets the expriation date of the object. Returns 0 if not known.

getDate

```
public long getDate()
```

Gets the sending date of the object. Returns 0 if not known.

getLastModified

```
public long getLastModified()
```

Gets the last modified date of the object. Returns 0 if not known.

getHeaderField

```
public String getHeaderField(String name)
```

Gets a header field by name. Returns null if not known.
Parameters:
name – the name of the header field

getHeaderFieldInt

```
public int getHeaderFieldInt(String name,
                             int Default)
```

Gets a header field by name. Returns null if not known. The field will be parsed as an integer. This form of getHeaderField exists because some connection types (eg. http–ng) have pre–parsed headers & this allows them to override this method and short–circuit the parsing.
Parameters:
name – the name of the header field
Default – the value to return if the field is missing or malformed.

getHeaderFieldDate

```
public long getHeaderFieldDate(String name,
                               long Default)
```

Gets a header field by name. Returns null if not known. The field will be parsed as a date. This form of getHeaderField exists because some connection types (eg. http–ng) have pre–parsed headers & this allows them to override this method and

short–circuit the parsing.

Parameters:

name – the name of the header field

Default – the value to return if the field is missing or malformed.

getHeaderFieldKey

```
public String getHeaderFieldKey(int n)
```

Return the key for the nth header field. Returns null if there are fewer than n fields. This can be used to iterate through all the headers in the message.

getHeaderField

```
public String getHeaderField(int n)
```

Return the value for the nth header field. Returns null if there are fewer than n fields. This can be used in conjunction with getHeaderFieldKey to iterate through all the headers in the message.

getContent

```
public Object getContent() throws IOException
```

Gets the object referred to by this URL. For example, if it refers to an image the object will be some subclass of Image. The instanceof operator should be used to determine what kind of object was returned.

Returns:

the object that was fetched.

Throws: UnknownServiceException

If the protocol does not support content.

getInputStream

```
public InputStream getInputStream() throws IOException
```

Calls this routine to get an InputStream that reads from the object. Protocol implementors should implement this if appropriate.

Throws: UnknownServiceException

If the protocol does not support input.

getOutputStream

```
public OutputStream getOutputStream() throws IOException
```

Calls this routine to get an OutputStream that writes to the object. Protocol implementors should implement this if appropriate.

Throws: UnknownServiceException

If the protocol does not support output.

toString

```
public String toString()
```

Returns the String representation of the URL connection.
Overrides:
toString in class Object

setDoInput

```
public void setDoInput(boolean doinput)
```

A URL connection can be used for input and/or output. Set the DoInput flag to true if you intend to use the URL connection for input, false if not. The default is true unless DoOutput is explicitly set to true, in which case DoInput defaults to false.

getDoInput

```
public boolean getDoInput()
```

setDoOutput

```
public void setDoOutput(boolean dooutput)
```

A URL connection can be used for input and/or output. Set the DoOutput flag to true if you intend to use the URL connection for output, false if not. The default is false.

getDoOutput

```
public boolean getDoOutput()
```

setAllowUserInteraction

```
public void setAllowUserInteraction(boolean allowuserinteraction)
```

Some URL connections occasionally need to to interactions with the user. For example, the http protocol may need to pop up an authentication dialog. But this is only appropriate if the application is running in a situation where there *is* a user. The allowUserInteraction flag allows these interactions when true. When it is false, they are not allowed an exception is tossed. The default value can be set/gotten using setDefaultAllowUserInteraction, which defaults to false.

getAllowUserInteraction

```
public boolean getAllowUserInteraction()
```

setDefaultAllowUserInteraction

```
public static void setDefaultAllowUserInteraction(boolean defaultallowuserinteraction)
```

Set/get the default value of the allowUserInteraction flag. This default is "sticky", being a part of the static state of all URLConnections. This flag applies to the next, and all following, URLConnections that are created.

getDefaultAllowUserInteraction

```
public static boolean getDefaultAllowUserInteraction()
```

setUseCaches

```
public void setUseCaches(boolean usecaches)
```

Some protocols do caching of documents. Occasionally, it is important to be able to "tunnel through" and ignore the caches (eg. the "reload" button in a browser). If the UseCaches flag on a connection is true, the connection is allowed to use whatever caches it can. If false, caches are to be ignored. The default value comes from DefaultUseCaches, which defaults to true.

getUseCaches

```
public boolean getUseCaches()
```

setIfModifiedSince

```
public void setIfModifiedSince(long ifmodifiedsince)
```

Some protocols support skipping fetching unless the object is newer than some time. The ifModifiedSince field may be set/gotten to define this time.

getIfModifiedSince

```
public long getIfModifiedSince()
```

getDefaultUseCaches

```
public boolean getDefaultUseCaches()
```

Set/get the default value of the UseCaches flag. This default is "sticky", being a part of the static state of all URLConnections. This flag applies to the next, and all following, URLConnections that are created.

setDefaultUseCaches

```
public void setDefaultUseCaches(boolean defaultusecaches)
```

setRequestProperty

```
public void setRequestProperty(String key,
                               String value)
```

Set/get a general request property.

Parameters:

key – The keyword by which the request is known (eg "accept")

value – The value associated with it.

getRequestProperty

```
public String getRequestProperty(String key)
```

setDefaultRequestProperty

```
public static void setDefaultRequestProperty(String key,
                                             String value)
```

Set/get the default value of a general request property. When a URLConnection is created, it gets initialized with these properties.

Parameters:

key – The keyword by which the request is known (eg "accept")

value – The value associated with it.

getDefaultRequestProperty

```
public static String getDefaultRequestProperty(String key)
```

setContentHandlerFactory

```
public static synchronized void setContentHandlerFactory(ContentHandlerFactory fac)
```

Sets the ContentHandler factory.

Parameters:

fac – the desired factory

Throws: Error

If the factory has already been defined.

guessContentTypeFromName

```
protected static String guessContentTypeFromName(String fname)
```

A useful utility routine that tries to guess the content–type of an object based upon its extension.

guessContentTypeFromStream

```
protected static String guessContentTypeFromStream(InputStream is) throws IOException
```

This disgusting hack is used to check for files have some type that can be

determined by inspection. The bytes at the beginning of the file are examined loosely. In an ideal world, this routine would not be needed, but in a world where http servers lie about content–types and extensions are often non–standard, direct inspection of the bytes can make the system more robust. The stream must support marks (eg. have a BufferedInputStream somewhere).

Class java.net.URLEncoder

```
java.lang.Object
   |
   +----java.net.URLEncoder
```

---

public class **URLEncoder**
extends Object

URLEncoder: turns Strings of text into x–www–form–urlencoded format.

Method Index

- **encode**(String)
  encode – translate String into x–www–form–urlencoded format.

Methods

encode

```
public static String encode(String s)
```

encode – translate String into x–www–form–urlencoded format.
Parameters:
s – String to be translated
Returns:
the translated String.

Class java.net.URLStreamHandler

```
java.lang.Object
   |
   +----java.net.URLStreamHandler
```

---

public class **URLStreamHandler**
extends Object

Abstract class for URL stream openers. Subclasses of this class know how to create streams for particular protocol types.

Constructor Index

- **URLStreamHandler**()

Method Index

- **openConnection**(URL)
  Opens an input stream to the object referenced by the URL.
- **parseURL**(URL, String, int, int)
  This method is called to parse the string spec into URL u.
- **setURL**(URL, String, String, int, String, String)
  Calls the (protected) set method out of the URL given.
- **toExternalForm**(URL)
  Reverses the parsing of the URL.

Constructors

URLStreamHandler

```
public URLStreamHandler()
```

Methods

openConnection

```
protected abstract URLConnection openConnection(URL u) throws IOException
```

Opens an input stream to the object referenced by the URL. This method should be overridden by a subclass.

Parameters:

u – the URL that this connects to

parseURL

```
protected void parseURL(URL u,
                        String spec,
                        int start,
                        int limit)
```

This method is called to parse the string spec into URL u. If there is any inherited context then it has already been copied into u. The parameters start and limit refer to the range of characters in spec that should be parsed. The default method uses parsing rules that match the http spec, which most URL protocol families follow. If you are writing a protocol handler that has a different syntax, then this routine should be overridden.

Parameters:

u – the URL to receive the result of parsing the spec
spec – the URL string to parse
start – the character position to start parsing at. This is just past the ':' (if there is one)
limit – the character position to stop parsing at. This is the end of the string or the position of the "#" character if present (the "#" reference syntax is protocol independant).

toExternalForm

```
protected String toExternalForm(URL u)
```

Reverses the parsing of the URL. This should probably be overridden if you override parseURL().

Parameters:

u – the URL

Returns:

the textual representation of the fully qualified URL (ie. after the context and canonicalization have been applied).

setURL

```
protected void setURL(URL u,
```

```
String protocol,
String host,
int port,
String file,
String ref)
```

Calls the (protected) set method out of the URL given. Only classes derived from URLStreamHandler are supposed to be able to call the set() method on a URL.

See Also:

set

Interface java.net.URLStreamHandlerFactory

public interface **URLStreamHandlerFactory**
extends Object

This interface defines a factory for URLStreamHandler instances. It is used by the URL class to create URLStreamHandlers for various streams.

Method Index

Methods

createURLStreamHandler

```
public abstract URLStreamHandler createURLStreamHandler(String protocol)
```

Creates a new URLStreamHandler instance with the specified protocol.
Parameters:
protocol – the protocol to use (ftp, http, nntp, etc.)

Class java.net.UnknownHostException

```
java.lang.Object
   |
   +----java.lang.Throwable
           |
           +----java.lang.Exception
                   |
                   +----java.io.IOException
                           |
                           +----java.net.UnknownHostException
```

---

public class **UnknownHostException**
extends IOException

Signals that the address of the server specified by a network client could not be resolved.

Constructor Index

- **UnknownHostException**(String)
  Constructs a new UnknownHostException with the specified detail message.
- **UnknownHostException**()
  Constructs a new UnknownHostException with no detail message.

Constructors

UnknownHostException

```
public UnknownHostException(String host)
```

Constructs a new UnknownHostException with the specified detail message. A detail message is a String that gives a specific description of this error.
Parameters:
host – the detail message

UnknownHostException

```
public UnknownHostException()
```

Constructs a new UnknownHostException with no detail message. A detail message is a String that gives a specific description of this error.

Class java.net.UnknownServiceException

```
java.lang.Object
   |
   +----java.lang.Throwable
           |
           +----java.lang.Exception
                   |
                   +----java.io.IOException
                           |
                           +----java.net.UnknownServiceException
```

---

public class **UnknownServiceException**
extends IOException

Signals that an unknown service exception has occurred.

Constructor Index

- **UnknownServiceException**()
  Constructs a new UnknownServiceException with no detail message.
- **UnknownServiceException**(String)
  Constructs a new UnknownServiceException with the specified detail message.

Constructors

UnknownServiceException

```
public UnknownServiceException()
```

Constructs a new UnknownServiceException with no detail message. A detail message is a String that gives a specific description of this error.

UnknownServiceException

```
public UnknownServiceException(String msg)
```

Constructs a new UnknownServiceException with the specified detail message. A detail message is a String that gives a specific description of this error.

Parameters:

msg – the detail message
