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.

Version:
    1.16, 10/24/95
Author:
    James Gosling

Variable Index

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

Constructor Index

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

Method Index

o **connect**()
    URLConnection objects go through two phases: first they are created, then they are connected.

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

Sets the ContentHandler factory.

o **setDefaultAllowUserInteraction**(boolean)
    Set/get the default value of the allowUserInteraction flag.

o **setDefaultRequestProperty**(String, String)
    Set/get the default value of a general request property.

o **setDefaultUseCaches**(boolean)

o **setDoInput**(boolean)
    A URL connection can be used for input and/or output.

o **setDoOutput**(boolean)
    A URL connection can be used for input and/or output.

o **setIfModifiedSince**(long)
    Some protocols support skipping fetching unless the object is newer than some time.

o **setRequestProperty**(String, String)
    Set/get a general request property.

o **setUseCaches**(boolean)
    Some protocols do caching of documents.

o **toString**()
    Returns the String representation of the URL connection.

Variables

o **url**

```
protected URL url
```

o **doInput**

```
protected boolean doInput
```

o **doOutput**

```
protected boolean doOutput
```

o **allowUserInteraction**

```
protected boolean allowUserInteraction
```

o **useCaches**

```
protected boolean useCaches
```

o **ifModifiedSince**

```
protected long ifModifiedSince
```

o **connected**

```
protected boolean connected
```

Constructors

o URLConnection

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

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

Methods

o 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.

o getURL

```
public URL getURL()
```

Gets the URL for this connection.

o getContentLength

```
public int getContentLength()
```

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

o getContentType

```
public String getContentType()
```

Gets the content type. Returns null if not known.

o getContentEncoding

```
public String getContentEncoding()
```

Gets the content encoding. Returns null if not known.

o getExpiration

```
public long getExpiration()
```

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

o **getDate**

```
public long getDate()
```

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

o **getLastModified**

```
public long getLastModified()
```

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

o **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

o **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.

o **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.

o **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.

o **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.

o **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.

o **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.

o **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.

o **toString**

```
public String toString()
```

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

o **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.

o **getDoInput**

```
public boolean getDoInput()
```

o **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.

o **getDoOutput**

```
public boolean getDoOutput()
```

o **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.

o **getAllowUserInteraction**

```
public boolean getAllowUserInteraction()
```

o **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.

o **getDefaultAllowUserInteraction**

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

o **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.

o **getUseCaches**

```
public boolean getUseCaches()
```

o **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.

o **getIfModifiedSince**

```
public long getIfModifiedSince()
```

o **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.

o **setDefaultUseCaches**

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

o **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.

o **getRequestProperty**

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

o **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.

o **getDefaultRequestProperty**

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

o **setContentHandlerFactory**

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

Sets the ContentHandler factory.
Parameters:
fac – the desired factory
Throws: Exception
If the factory has already been defined.

o **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.

o **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).
