java.net

Class URLConnection

java.lang.Object
    java.net.URLConnection

Direct Known Subclasses:

`HttpURLConnection`, `JarURLConnection`

```
public abstract class URLConnection
extends Object
```

The abstract class `URLConnection` is the superclass of all classes that represent a communications link between the application and a URL. Instances of this class can be used both to read from and to write to the resource referenced by the URL. In general, creating a connection to a URL is a multistep process:

`openConnection()`	`connect()`
Manipulate parameters that affect the connection to the remote resource.	Interact with the resource; query header fields and contents.

----------->

time

1. The connection object is created by invoking the `openConnection` method on a URL.
2. The setup parameters and general request properties are manipulated.
3. The actual connection to the remote object is made, using the `connect` method.
4. The remote object becomes available. The header fields and the contents of the remote object can be accessed.

The setup parameters are modified using the following methods:

- `setAllowUserInteraction`
- `setDoInput`
- `setDoOutput`
- `setIfModifiedSince`
- `setUseCaches`

and the general request properties are modified using the method:

- `setRequestProperty`

Default values for the `AllowUserInteraction` and `UseCaches` parameters can be set using the methods `setDefaultAllowUserInteraction` and `setDefaultUseCaches`.

Each of the above `set` methods has a corresponding `get` method to retrieve the value of the parameter or general request property. The specific parameters and general request properties that are applicable are protocol specific.

The following methods are used to access the header fields and the contents after the connection is made to the remote object:

- `getContent`

- getContentEncoding
- getContentLength
- getContentType
- getDate
- getExpiration
- getLastModifed

provide convenient access to these fields. The `getContentType` method is used by the `getContent` method to determine the type of the remote object; subclasses may find it convenient to override the `getContentType` method.

In the common case, all of the pre-connection parameters and general request properties can be ignored: the pre-connection parameters and request properties default to sensible values. For most clients of this interface, there are only two interesting methods: `getInputStream` and `getContent`, which are mirrored in the `URL` class by convenience methods.

More information on the request properties and header fields of an `http` connection can be found at:

> http://www.ietf.org/rfc/rfc2616.txt

Invoking the `close()` methods on the `InputStream` or `OutputStream` of an `URLConnection` after a request may free network resources associated with this instance, unless particular protocol specifications specify different behaviours for it.

Since:

JDK1.0

See Also:

`URL.openConnection()`, `connect()`, `getContent()`, `getContentEncoding()`, `getContentLength()`, `getContentType()`, `getDate()`, `getExpiration()`, `getHeaderField(int)`, `getHeaderField(java.lang.String)`, `getInputStream()`, `getLastModified()`, `getOutputStream()`, `setAllowUserInteraction(boolean)`, `setDefaultUseCaches(boolean)`, `setDoInput(boolean)`, `setDoOutput(boolean)`, `setIfModifiedSince(long)`, `setRequestProperty(java.lang.String, java.lang.String)`, `setUseCaches(boolean)`

Field Summary

Fields

Modifier and Type	Field and Description
protected boolean	**allowUserInteraction** If `true`, this `URL` is being examined in a context in which it makes sense to allow user interactions such as popping up an authentication dialog.
protected boolean	**connected** If `false`, this connection object has not created a communications link to the specified URL.
protected boolean	**doInput**

	Some protocols support skipping the fetching of the object unless the object has been modified more recently than a certain time.
protected URL	**url** The URL represents the remote object on the World Wide Web to which this connection is opened.
protected boolean	**useCaches** If `true`, the protocol is allowed to use caching whenever it can.

Constructor Summary

Constructors

Modifier	Constructor and Description
protected	**URLConnection**(**URL** url) Constructs a URL connection to the specified URL.

Method Summary

All Methods Static Methods Instance Methods Abstract Methods Concrete Methods Deprecated Methods

Modifier and Type	Method and Description
void	**addRequestProperty**(**String** key, **String** value) Adds a general request property specified by a key-value pair.
abstract void	**connect**() Opens a communications link to the resource referenced by this URL, if such a connection has not already been established.
boolean	**getAllowUserInteraction**() Returns the value of the `allowUserInteraction` field for this object.
int	**getConnectTimeout**() Returns setting for connect timeout.
Object	**getContent**() Retrieves the contents of this URL connection.
Object	**getContent**(**Class**[] classes) Retrieves the contents of this URL connection.
String	**getContentEncoding**() Returns the value of the `content-encoding` header field.
int	**getContentLength**()

Returns the value of the `content-type` header field.

`long`	**`getDate`**`()` Returns the value of the `date` header field.
`static boolean`	**`getDefaultAllowUserInteraction`**`()` Returns the default value of the `allowUserInteraction` field.
`static String`	**`getDefaultRequestProperty`**`(String key)` **Deprecated.** The instance specific getRequestProperty method should be used after an appropriate instance of URLConnection is obtained.
`boolean`	**`getDefaultUseCaches`**`()` Returns the default value of a `URLConnection`'s `useCaches` flag.
`boolean`	**`getDoInput`**`()` Returns the value of this `URLConnection`'s `doInput` flag.
`boolean`	**`getDoOutput`**`()` Returns the value of this `URLConnection`'s `doOutput` flag.
`long`	**`getExpiration`**`()` Returns the value of the `expires` header field.
`static FileNameMap`	**`getFileNameMap`**`()` Loads filename map (a mimetable) from a data file.
`String`	**`getHeaderField`**`(int n)` Returns the value for the nth header field.
`String`	**`getHeaderField`**`(String name)` Returns the value of the named header field.
`long`	**`getHeaderFieldDate`**`(String name, long Default)` Returns the value of the named field parsed as date.
`int`	**`getHeaderFieldInt`**`(String name, int Default)` Returns the value of the named field parsed as a number.
`String`	**`getHeaderFieldKey`**`(int n)` Returns the key for the nth header field.
`long`	**`getHeaderFieldLong`**`(String name, long Default)` Returns the value of the named field parsed as a number.
`Map<String,List<String>>`	**`getHeaderFields`**`()` Returns an unmodifiable Map of the header fields.
`long`	**`getIfModifiedSince`**`()` Returns the value of this object's `ifModifiedSince` field.

OutputStream	**getOutputStream**() Returns an output stream that writes to this connection.
Permission	**getPermission**() Returns a permission object representing the permission necessary to make the connection represented by this object.
int	**getReadTimeout**() Returns setting for read timeout.
Map<String,List<String>>	**getRequestProperties**() Returns an unmodifiable Map of general request properties for this connection.
String	**getRequestProperty**(String key) Returns the value of the named general request property for this connection.
URL	**getURL**() Returns the value of this URLConnection's URL field.
boolean	**getUseCaches**() Returns the value of this URLConnection's useCaches field.
static String	**guessContentTypeFromName**(String fname) Tries to determine the content type of an object, based on the specified "file" component of a URL.
static String	**guessContentTypeFromStream**(InputStream is) Tries to determine the type of an input stream based on the characters at the beginning of the input stream.
void	**setAllowUserInteraction**(boolean allowuserinteraction) Set the value of the allowUserInteraction field of this URLConnection.
void	**setConnectTimeout**(int timeout) Sets a specified timeout value, in milliseconds, to be used when opening a communications link to the resource referenced by this URLConnection.
static void	**setContentHandlerFactory**(ContentHandlerFactory fac) Sets the ContentHandlerFactory of an application.
static void	**setDefaultAllowUserInteraction**(boolean defaultallowuserinteraction) Sets the default value of the allowUserInteraction field for all future URLConnection objects to the specified value.
static void	**setDefaultRequestProperty**(String key, String value) **Deprecated.** The instance specific setRequestProperty method should be used after an appropriate instance of URLConnection is obtained. Invoking this method will have no effect.

void	**setDoOutput**(boolean dooutput) Sets the value of the doOutput field for this URLConnection to the specified value.
static void	**setFileNameMap**(**FileNameMap** map) Sets the FileNameMap.
void	**setIfModifiedSince**(long ifmodifiedsince) Sets the value of the ifModifiedSince field of this URLConnection to the specified value.
void	**setReadTimeout**(int timeout) Sets the read timeout to a specified timeout, in milliseconds.
void	**setRequestProperty**(**String** key, **String** value) Sets the general request property.
void	**setUseCaches**(boolean usecaches) Sets the value of the useCaches field of this URLConnection to the specified value.
String	**toString**() Returns a String representation of this URL connection.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

url

```
protected URL url
```

The URL represents the remote object on the World Wide Web to which this connection is opened.

The value of this field can be accessed by the getURL method.

The default value of this variable is the value of the URL argument in the URLConnection constructor.

See Also:

getURL(), url

doInput

```
protected boolean doInput
```

See Also:

`getDoInput()`, `setDoInput(boolean)`

doOutput

```
protected boolean doOutput
```

This variable is set by the `setDoOutput` method. Its value is returned by the `getDoOutput` method.

A URL connection can be used for input and/or output. Setting the `doOutput` flag to `true` indicates that the application intends to write data to the URL connection.

The default value of this field is `false`.

See Also:

`getDoOutput()`, `setDoOutput(boolean)`

allowUserInteraction

```
protected boolean allowUserInteraction
```

If `true`, this `URL` is being examined in a context in which it makes sense to allow user interactions such as popping up an authentication dialog. If `false`, then no user interaction is allowed.

The value of this field can be set by the `setAllowUserInteraction` method. Its value is returned by the `getAllowUserInteraction` method. Its default value is the value of the argument in the last invocation of the `setDefaultAllowUserInteraction` method.

See Also:

`getAllowUserInteraction()`, `setAllowUserInteraction(boolean)`, `setDefaultAllowUserInteraction(boolean)`

useCaches

```
protected boolean useCaches
```

If `true`, the protocol is allowed to use caching whenever it can. If `false`, the protocol must always try to get a fresh copy of the object.

This field is set by the `setUseCaches` method. Its value is returned by the `getUseCaches` method.

Its default value is the value given in the last invocation of the `setDefaultUseCaches` method.

See Also:

`setUseCaches(boolean)`, `getUseCaches()`, `setDefaultUseCaches(boolean)`

ifModifiedSince

fetched only if it has been modified more recently than that time.

This variable is set by the `setIfModifiedSince` method. Its value is returned by the `getIfModifiedSince` method.

The default value of this field is `0`, indicating that the fetching must always occur.

See Also:

`getIfModifiedSince()`, `setIfModifiedSince(long)`

connected

```
protected boolean connected
```

If `false`, this connection object has not created a communications link to the specified URL. If `true`, the communications link has been established.

Constructor Detail

URLConnection

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

Constructs a URL connection to the specified URL. A connection to the object referenced by the URL is not created.

Parameters:

`url - the specified URL.`

Method Detail

getFileNameMap

```
public static FileNameMap getFileNameMap()
```

Loads filename map (a mimetable) from a data file. It will first try to load the user-specific table, defined by "content.types.user.table" property. If that fails, it tries to load the default built-in table.

Returns:

`the FileNameMap`

Since:

`1.2`

See Also:

Sets the FileNameMap.

If there is a security manager, this method first calls the security manager's `checkSetFactory` method to ensure the operation is allowed. This could result in a SecurityException.

Parameters:

`map - the FileNameMap to be set`

Throws:

`SecurityException - if a security manager exists and its checkSetFactory method doesn't allow the operation.`

Since:

`1.2`

See Also:

`SecurityManager.checkSetFactory(), getFileNameMap()`

connect

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

Opens a communications link to the resource referenced by this URL, if such a connection has not already been established.

If the `connect` method is called when the connection has already been opened (indicated by the `connected` field having the value `true`), the call is ignored.

URLConnection objects go through two phases: first they are created, then they are connected. After being created, and before being connected, various options can be specified (e.g., doInput and 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.

Throws:

`SocketTimeoutException - if the timeout expires before the connection can be established`

`IOException - if an I/O error occurs while opening the connection.`

See Also:

`connected, getConnectTimeout(), setConnectTimeout(int)`

setConnectTimeout

```
public void setConnectTimeout(int timeout)
```

Sets a specified timeout value, in milliseconds, to be used when opening a communications link to the resource referenced by this URLConnection. If the timeout expires before the connection can be

Parameters:

`timeout - an int that specifies the connect timeout value in milliseconds`

Throws:

`IllegalArgumentException - if the timeout parameter is negative`

Since:

1.5

See Also:

`getConnectTimeout()`, `connect()`

getConnectTimeout

```
public int getConnectTimeout()
```

Returns setting for connect timeout.

0 return implies that the option is disabled (i.e., timeout of infinity).

Returns:

`an int that indicates the connect timeout value in milliseconds`

Since:

1.5

See Also:

`setConnectTimeout(int)`, `connect()`

setReadTimeout

```
public void setReadTimeout(int timeout)
```

Sets the read timeout to a specified timeout, in milliseconds. A non-zero value specifies the timeout when reading from Input stream when a connection is established to a resource. If the timeout expires before there is data available for read, a java.net.SocketTimeoutException is raised. A timeout of zero is interpreted as an infinite timeout.

Some non-standard implementation of this method ignores the specified timeout. To see the read timeout set, please call getReadTimeout().

Parameters:

`timeout - an int that specifies the timeout value to be used in milliseconds`

Throws:

`IllegalArgumentException - if the timeout parameter is negative`

Since:

1.5

See Also:

Returns setting for read timeout. 0 return implies that the option is disabled (i.e., timeout of infinity).

Returns:

an int that indicates the read timeout value in milliseconds

Since:

1.5

See Also:

`setReadTimeout(int)`, `InputStream.read()`

getURL

```
public URL getURL()
```

Returns the value of this `URLConnection`'s `URL` field.

Returns:

the value of this URLConnection's URL field.

See Also:

`url`

getContentLength

```
public int getContentLength()
```

Returns the value of the `content-length` header field.

Note: `getContentLengthLong()` should be preferred over this method, since it returns a `long` instead and is therefore more portable.

Returns:

the content length of the resource that this connection's URL references, -1 if the content length is not known, or if the content length is greater than Integer.MAX_VALUE.

getContentLengthLong

```
public long getContentLengthLong()
```

Returns the value of the `content-length` header field as a long.

Returns:

the content length of the resource that this connection's URL references, or -1 if the content length is not known.

Since:

Returns the value of the `content-type` header field.

Returns:

the content type of the resource that the URL references, or null if not known.

See Also:

`getHeaderField(java.lang.String)`

getContentEncoding

```
public String getContentEncoding()
```

Returns the value of the `content-encoding` header field.

Returns:

the content encoding of the resource that the URL references, or null if not known.

See Also:

`getHeaderField(java.lang.String)`

getExpiration

```
public long getExpiration()
```

Returns the value of the `expires` header field.

Returns:

the expiration date of the resource that this URL references, or 0 if not known. The value is the number of milliseconds since January 1, 1970 GMT.

See Also:

`getHeaderField(java.lang.String)`

getDate

```
public long getDate()
```

Returns the value of the `date` header field.

Returns:

the sending date of the resource that the URL references, or 0 if not known. The value returned is the number of milliseconds since January 1, 1970 GMT.

See Also:

`getHeaderField(java.lang.String)`

Returns:

the date the resource referenced by this URLConnection was last modified, or 0 if not known.

See Also:

getHeaderField(java.lang.String)

getHeaderField

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

Returns the value of the named header field.

If called on a connection that sets the same header multiple times with possibly different values, only the last value is returned.

Parameters:

name - the name of a header field.

Returns:

the value of the named header field, or null if there is no such field in the header.

getHeaderFields

```
public Map<String,List<String>> getHeaderFields()
```

Returns an unmodifiable Map of the header fields. The Map keys are Strings that represent the response-header field names. Each Map value is an unmodifiable List of Strings that represents the corresponding field values.

Returns:

a Map of header fields

Since:

1.4

getHeaderFieldInt

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

Returns the value of the named field parsed as a number.

This form of `getHeaderField` exists because some connection types (e.g., `http-ng`) have pre-parsed headers. Classes for that connection type can override this method and short-circuit the parsing.

Parameters:

getHeaderFieldLong

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

Returns the value of the named field parsed as a number.

This form of `getHeaderField` exists because some connection types (e.g., `http-ng`) have pre-parsed headers. Classes for that connection type can override this method and short-circuit the parsing.

Parameters:

`name - the name of the header field.`

`Default - the default value.`

Returns:

`the value of the named field, parsed as a long. The Default value is returned if the field is missing or malformed.`

Since:

`7.0`

getHeaderFieldDate

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

Returns the value of the named field parsed as date. The result is the number of milliseconds since January 1, 1970 GMT represented by the named field.

This form of `getHeaderField` exists because some connection types (e.g., `http-ng`) have pre-parsed headers. Classes for that connection type can override this method and short-circuit the parsing.

Parameters:

`name - the name of the header field.`

`Default - a default value.`

Returns:

`the value of the field, parsed as a date. The value of the Default argument is returned if the field is missing or malformed.`

getHeaderFieldKey

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

Returns the key for the n^{th} header field. It returns `null` if there are fewer than `n+1` fields.

Parameters:

getHeaderField

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

Returns the value for the n^{th} header field. It returns `null` if there are fewer than `n+1`fields.

This method can be used in conjunction with the `getHeaderFieldKey` method to iterate through all the headers in the message.

Parameters:

`n - an index, where n>=0`

Returns:

`the value of the n^th header field or null if there are fewer than n+1 fields`

See Also:

`getHeaderFieldKey(int)`

getContent

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

Retrieves the contents of this URL connection.

This method first determines the content type of the object by calling the `getContentType` method. If this is the first time that the application has seen that specific content type, a content handler for that content type is created:

1. If the application has set up a content handler factory instance using the `setContentHandlerFactory` method, the `createContentHandler` method of that instance is called with the content type as an argument; the result is a content handler for that content type.
2. If no content handler factory has yet been set up, or if the factory's `createContentHandler` method returns `null`, then the application loads the class named:

   `sun.net.www.content.<contentType>`

   where *<contentType>* is formed by taking the content-type string, replacing all slash characters with a `period` ('.'), and all other non-alphanumeric characters with the underscore character '_'. The alphanumeric characters are specifically the 26 uppercase ASCII letters 'A' through 'Z', the 26 lowercase ASCII letters 'a' through 'z', and the 10 ASCII digits '0' through '9'. If the specified class does not exist, or is not a subclass of `ContentHandler`, then an `UnknownServiceException` is thrown.

Returns:

`the object fetched. The instanceof operator should be used to determine the specific kind of object returned.`

Throws:

`IOException - if an I/O error occurs while getting the content.`

getContent

```
public Object getContent(Class[] classes)
                  throws IOException
```

Retrieves the contents of this URL connection.

Parameters:

`classes - the Class array indicating the requested types`

Returns:

`the object fetched that is the first match of the type specified in the classes array. null if none of the requested types are supported. The instanceof operator should be used to determine the specific kind of object returned.`

Throws:

`IOException - if an I/O error occurs while getting the content.`

`UnknownServiceException - if the protocol does not support the content type.`

Since:

`1.3`

See Also:

`getContent(), ContentHandlerFactory.createContentHandler(java.lang.String), getContent(java.lang.Class[]), setContentHandlerFactory(java.net.ContentHandlerFactory)`

getPermission

```
public Permission getPermission()
                        throws IOException
```

Returns a permission object representing the permission necessary to make the connection represented by this object. This method returns null if no permission is required to make the connection. By default, this method returns `java.security.AllPermission`. Subclasses should override this method and return the permission that best represents the permission required to make a a connection to the URL. For example, a `URLConnection` representing a `file:` URL would return a `java.io.FilePermission` object.

The permission returned may dependent upon the state of the connection. For example, the permission before connecting may be different from that after connecting. For example, an HTTP sever, say foo.com, may redirect the connection to a different host, say bar.com. Before connecting the permission returned by the connection will represent the permission needed to connect to foo.com, while the permission returned after connecting will be to bar.com.

Permissions are generally used for two purposes: to protect caches of objects obtained through URLConnections, and to check the right of a recipient to learn about a particular URL. In the first case, the permission should be obtained *after* the object has been obtained. For example, in an HTTP connection, this will represent the permission to connect to the host from which the data was

Throws:

IOException - if the computation of the permission requires network or file I/O and an exception occurs while computing it.

getInputStream

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

Returns an input stream that reads from this open connection. A SocketTimeoutException can be thrown when reading from the returned input stream if the read timeout expires before data is available for read.

Returns:

an input stream that reads from this open connection.

Throws:

IOException - if an I/O error occurs while creating the input stream.

UnknownServiceException - if the protocol does not support input.

See Also:

setReadTimeout(int), getReadTimeout()

getOutputStream

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

Returns an output stream that writes to this connection.

Returns:

an output stream that writes to this connection.

Throws:

IOException - if an I/O error occurs while creating the output stream.

UnknownServiceException - if the protocol does not support output.

toString

```
public String toString()
```

Returns a String representation of this URL connection.

Overrides:

toString in class Object

Returns:

Sets the value of the `doInput` field for this `URLConnection` to the specified value.

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.

Parameters:

`doinput - the new value.`

Throws:

`IllegalStateException - if already connected`

See Also:

`doInput, getDoInput()`

getDoInput

```
public boolean getDoInput()
```

Returns the value of this `URLConnection`'s `doInput` flag.

Returns:

`the value of this URLConnection's doInput flag.`

See Also:

`setDoInput(boolean)`

setDoOutput

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

Sets the value of the `doOutput` field for this `URLConnection` to the specified value.

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.

Parameters:

`dooutput - the new value.`

Throws:

`IllegalStateException - if already connected`

See Also:

`getDoOutput()`

getDoOutput

```
public boolean getDoOutput()
```

Returns the value of this `URLConnection`'s `doOutput` flag.

setAllowUserInteraction

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

Set the value of the `allowUserInteraction` field of this `URLConnection`.

Parameters:

`allowuserinteraction - the new value.`

Throws:

`IllegalStateException - if already connected`

See Also:

`getAllowUserInteraction()`

getAllowUserInteraction

```
public boolean getAllowUserInteraction()
```

Returns the value of the `allowUserInteraction` field for this object.

Returns:

`the value of the allowUserInteraction field for this object.`

See Also:

`setAllowUserInteraction(boolean)`

setDefaultAllowUserInteraction

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

Sets the default value of the `allowUserInteraction` field for all future `URLConnection` objects to the specified value.

Parameters:

`defaultallowuserinteraction - the new value.`

See Also:

`getDefaultAllowUserInteraction()`

getDefaultAllowUserInteraction

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

Returns the default value of the `allowUserInteraction` field.

Ths 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.

setUseCaches

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

Sets the value of the `useCaches` field of this `URLConnection` to the specified value.

Some protocols do caching of documents. Occasionally, it is important to be able to "tunnel through" and ignore the caches (e.g., 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.

Parameters:

`usecaches - a boolean indicating whether or not to allow caching`

Throws:

`IllegalStateException - if already connected`

See Also:

`getUseCaches()`

getUseCaches

```
public boolean getUseCaches()
```

Returns the value of this `URLConnection`'s `useCaches` field.

Returns:

`the value of this URLConnection's useCaches field.`

See Also:

`setUseCaches(boolean)`

setIfModifiedSince

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

Sets the value of the `ifModifiedSince` field of this `URLConnection` to the specified value.

Parameters:

`ifmodifiedsince - the new value.`

Throws:

`IllegalStateException - if already connected`

See Also:

`getIfModifiedSince()`

getIfModifiedSince

See Also:

`setIfModifiedSince(long)`

getDefaultUseCaches

```
public boolean getDefaultUseCaches()
```

Returns the default value of a `URLConnection`'s `useCaches` flag.

Ths 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.

Returns:

`the default value of a URLConnection's useCaches flag.`

See Also:

`setDefaultUseCaches(boolean)`

setDefaultUseCaches

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

Sets the default value of the `useCaches` field to the specified value.

Parameters:

`defaultusecaches - the new value.`

See Also:

`getDefaultUseCaches()`

setRequestProperty

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

Sets the general request property. If a property with the key already exists, overwrite its value with the new value.

NOTE: HTTP requires all request properties which can legally have multiple instances with the same key to use a comma-separated list syntax which enables multiple properties to be appended into a single property.

Parameters:

`key - the keyword by which the request is known (e.g., "Accept").`

`value - the value associated with it.`

Throws:

`IllegalStateException - if already connected`

addRequestProperty

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

Adds a general request property specified by a key-value pair. This method will not overwrite existing values associated with the same key.

Parameters:

`key - the keyword by which the request is known (e.g., "Accept").`

`value - the value associated with it.`

Throws:

`IllegalStateException - if already connected`

`NullPointerException - if key is null`

Since:

`1.4`

See Also:

`getRequestProperties()`

getRequestProperty

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

Returns the value of the named general request property for this connection.

Parameters:

`key - the keyword by which the request is known (e.g., "Accept").`

Returns:

`the value of the named general request property for this connection. If key is null, then null is returned.`

Throws:

`IllegalStateException - if already connected`

See Also:

`setRequestProperty(java.lang.String, java.lang.String)`

getRequestProperties

```
public Map<String,List<String>> getRequestProperties()
```

Returns an unmodifiable Map of general request properties for this connection. The Map keys are Strings that represent the request-header field names. Each Map value is a unmodifiable List of Strings that represents the corresponding field values.

Returns:

setDefaultRequestProperty

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

Deprecated. *The instance specific setRequestProperty method should be used after an appropriate instance of URLConnection is obtained. Invoking this method will have no effect.*

Sets the default value of a general request property. When a `URLConnection` is created, it is initialized with these properties.

Parameters:

`key` - the keyword by which the request is known (e.g., "Accept").

`value` - the value associated with the key.

See Also:

`setRequestProperty(java.lang.String,java.lang.String)`, `getDefaultRequestProperty(java.lang.String)`

getDefaultRequestProperty

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

Deprecated. *The instance specific getRequestProperty method should be used after an appropriate instance of URLConnection is obtained.*

Returns the value of the default request property. Default request properties are set for every connection.

Parameters:

`key` - the keyword by which the request is known (e.g., "Accept").

Returns:

the value of the default request property for the specified key.

See Also:

`getRequestProperty(java.lang.String)`, `setDefaultRequestProperty(java.lang.String, java.lang.String)`

setContentHandlerFactory

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

Sets the `ContentHandlerFactory` of an application. It can be called at most once by an application.

The `ContentHandlerFactory` instance is used to construct a content handler from a content type

Throws:

`Error` - if the factory has already been defined.

`SecurityException` - if a security manager exists and its checkSetFactory method doesn't allow the operation.

See Also:

`ContentHandlerFactory`, `getContent()`, `SecurityManager.checkSetFactory()`

guessContentTypeFromName

```
public static String guessContentTypeFromName(String fname)
```

Tries to determine the content type of an object, based on the specified "file" component of a URL. This is a convenience method that can be used by subclasses that override the `getContentType` method.

Parameters:

`fname` - a filename.

Returns:

a guess as to what the content type of the object is, based upon its file name.

See Also:

`getContentType()`

guessContentTypeFromStream

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

Tries to determine the type of an input stream based on the characters at the beginning of the input stream. This method can be used by subclasses that override the `getContentType` method.

Ideally, this routine would not be needed. But many `http` servers return the incorrect content type; in addition, there are many nonstandard extensions. Direct inspection of the bytes to determine the content type is often more accurate than believing the content type claimed by the `http` server.

Parameters:

`is` - an input stream that supports marks.

Returns:

a guess at the content type, or null if none can be determined.

Throws:

`IOException` - if an I/O error occurs while reading the input stream.

See Also:

`InputStream.mark(int)`, `InputStream.markSupported()`, `getContentType()`