| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/URI.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/net/UnknownServiceException.html)   [**NEXT CLASS**](http://docs.google.com/java/net/URISyntaxException.html) | [**FRAMES**](http://docs.google.com/index.html?java/net/URI.html)    [**NO FRAMES**](http://docs.google.com/URI.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#1t3h5sf) | [METHOD](#4d34og8) | DETAIL: FIELD | [CONSTR](#17dp8vu) | [METHOD](#44sinio) |

## **java.net**

Class URI

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 **java.net.URI**

**All Implemented Interfaces:** [Serializable](http://docs.google.com/java/io/Serializable.html), [Comparable](http://docs.google.com/java/lang/Comparable.html)<[URI](http://docs.google.com/java/net/URI.html)>

public final class **URI**extends [Object](http://docs.google.com/java/lang/Object.html)implements [Comparable](http://docs.google.com/java/lang/Comparable.html)<[URI](http://docs.google.com/java/net/URI.html)>, [Serializable](http://docs.google.com/java/io/Serializable.html)

Represents a Uniform Resource Identifier (URI) reference.

Aside from some minor deviations noted below, an instance of this class represents a URI reference as defined by [*RFC 2396: Uniform Resource Identifiers (URI): Generic Syntax*](http://www.ietf.org/rfc/rfc2396.txt), amended by [*RFC 2732: Format for Literal IPv6 Addresses in URLs*](http://www.ietf.org/rfc/rfc2732.txt). The Literal IPv6 address format also supports scope\_ids. The syntax and usage of scope\_ids is described [here](http://docs.google.com/Inet6Address.html#scoped). This class provides constructors for creating URI instances from their components or by parsing their string forms, methods for accessing the various components of an instance, and methods for normalizing, resolving, and relativizing URI instances. Instances of this class are immutable.

#### URI syntax and components

At the highest level a URI reference (hereinafter simply "URI") in string form has the syntax[*scheme***:**]*scheme-specific-part*[**#***fragment*]where square brackets [...] delineate optional components and the characters **:** and **#** stand for themselves.

An *absolute* URI specifies a scheme; a URI that is not absolute is said to be *relative*. URIs are also classified according to whether they are *opaque* or *hierarchical*.

An *opaque* URI is an absolute URI whose scheme-specific part does not begin with a slash character ('/'). Opaque URIs are not subject to further parsing. Some examples of opaque URIs are:

| mailto:java-net@java.sun.com |  |
| --- | --- |
| news:comp.lang.java |  |
| urn:isbn:096139210x |

A *hierarchical* URI is either an absolute URI whose scheme-specific part begins with a slash character, or a relative URI, that is, a URI that does not specify a scheme. Some examples of hierarchical URIs are:

http://java.sun.com/j2se/1.3/

docs/guide/collections/designfaq.html#28

../../../demo/jfc/SwingSet2/src/SwingSet2.java

file:///~/calendar

A hierarchical URI is subject to further parsing according to the syntax

[*scheme***:**][**//***authority*][*path*][**?***query*][**#***fragment*]where the characters **:**, **/**, **?**, and **#** stand for themselves. The scheme-specific part of a hierarchical URI consists of the characters between the scheme and fragment components.

The authority component of a hierarchical URI is, if specified, either *server-based* or *registry-based*. A server-based authority parses according to the familiar syntax

[*user-info***@**]*host*[**:***port*]where the characters **@** and **:** stand for themselves. Nearly all URI schemes currently in use are server-based. An authority component that does not parse in this way is considered to be registry-based.

The path component of a hierarchical URI is itself said to be absolute if it begins with a slash character ('/'); otherwise it is relative. The path of a hierarchical URI that is either absolute or specifies an authority is always absolute.

All told, then, a URI instance has the following nine components:

| *Component* | *Type* |
| --- | --- |
| scheme | String |
| scheme-specific-part | String |
| authority | String |
| user-info | String |
| host | String |
| port | int |
| path | String |
| query | String |
| fragment | String |

In a given instance any particular component is either *undefined* or *defined* with a distinct value. Undefined string components are represented by null, while undefined integer components are represented by -1. A string component may be defined to have the empty string as its value; this is not equivalent to that component being undefined.

Whether a particular component is or is not defined in an instance depends upon the type of the URI being represented. An absolute URI has a scheme component. An opaque URI has a scheme, a scheme-specific part, and possibly a fragment, but has no other components. A hierarchical URI always has a path (though it may be empty) and a scheme-specific-part (which at least contains the path), and may have any of the other components. If the authority component is present and is server-based then the host component will be defined and the user-information and port components may be defined.

#### Operations on URI instances

The key operations supported by this class are those of *normalization*, *resolution*, and *relativization*.

*Normalization* is the process of removing unnecessary "." and ".." segments from the path component of a hierarchical URI. Each "." segment is simply removed. A ".." segment is removed only if it is preceded by a non-".." segment. Normalization has no effect upon opaque URIs.

*Resolution* is the process of resolving one URI against another, *base* URI. The resulting URI is constructed from components of both URIs in the manner specified by RFC 2396, taking components from the base URI for those not specified in the original. For hierarchical URIs, the path of the original is resolved against the path of the base and then normalized. The result, for example, of resolving

docs/guide/collections/designfaq.html#28          (1)against the base URI http://java.sun.com/j2se/1.3/ is the result URIhttp://java.sun.com/j2se/1.3/docs/guide/collections/designfaq.html#28Resolving the relative URI../../../demo/jfc/SwingSet2/src/SwingSet2.java    (2)against this result yields, in turn,http://java.sun.com/j2se/1.3/demo/jfc/SwingSet2/src/SwingSet2.javaResolution of both absolute and relative URIs, and of both absolute and relative paths in the case of hierarchical URIs, is supported. Resolving the URI file:///~calendar against any other URI simply yields the original URI, since it is absolute. Resolving the relative URI (2) above against the relative base URI (1) yields the normalized, but still relative, URIdemo/jfc/SwingSet2/src/SwingSet2.java

*Relativization*, finally, is the inverse of resolution: For any two normalized URIs *u* and *v*,

*u*.relativize(*u*.resolve(*v*)).equals(*v*)  and

*u*.resolve(*u*.relativize(*v*)).equals(*v*)  .This operation is often useful when constructing a document containing URIs that must be made relative to the base URI of the document wherever possible. For example, relativizing the URIhttp://java.sun.com/j2se/1.3/docs/guide/index.htmlagainst the base URIhttp://java.sun.com/j2se/1.3yields the relative URI docs/guide/index.html.

#### Character categories

RFC 2396 specifies precisely which characters are permitted in the various components of a URI reference. The following categories, most of which are taken from that specification, are used below to describe these constraints:

| *alpha* | The US-ASCII alphabetic characters, 'A' through 'Z' and 'a' through 'z' |
| --- | --- |
| *digit* | The US-ASCII decimal digit characters, '0' through '9' |
| *alphanum* | All *alpha* and *digit* characters |
| *unreserved* | All *alphanum* characters together with those in the string "\_-!.~'()\*" |
| *punct* | The characters in the string ",;:$&+=" |
| *reserved* | All *punct* characters together with those in the string "?/[]@" |
| *escaped* | Escaped octets, that is, triplets consisting of the percent character ('%') followed by two hexadecimal digits ('0'-'9', 'A'-'F', and 'a'-'f') |
| *other* | The Unicode characters that are not in the US-ASCII character set, are not control characters (according to the [Character.isISOControl](http://docs.google.com/java/lang/Character.html#isISOControl(char)) method), and are not space characters (according to the [Character.isSpaceChar](http://docs.google.com/java/lang/Character.html#isSpaceChar(char)) method)  *(****Deviation from RFC 2396****, which is limited to US-ASCII)* |

The set of all legal URI characters consists of the *unreserved*, *reserved*, *escaped*, and *other* characters.

#### Escaped octets, quotation, encoding, and decoding

RFC 2396 allows escaped octets to appear in the user-info, path, query, and fragment components. Escaping serves two purposes in URIs:

* To *encode* non-US-ASCII characters when a URI is required to conform strictly to RFC 2396 by not containing any *other* characters.
* To *quote* characters that are otherwise illegal in a component. The user-info, path, query, and fragment components differ slightly in terms of which characters are considered legal and illegal.

These purposes are served in this class by three related operations:

* A character is *encoded* by replacing it with the sequence of escaped octets that represent that character in the UTF-8 character set. The Euro currency symbol ('\u20AC'), for example, is encoded as "%E2%82%AC". *(****Deviation from RFC 2396****, which does not specify any particular character set.)*

* An illegal character is *quoted* simply by encoding it. The space character, for example, is quoted by replacing it with "%20". UTF-8 contains US-ASCII, hence for US-ASCII characters this transformation has exactly the effect required by RFC 2396.

* A sequence of escaped octets is *decoded* by replacing it with the sequence of characters that it represents in the UTF-8 character set. UTF-8 contains US-ASCII, hence decoding has the effect of de-quoting any quoted US-ASCII characters as well as that of decoding any encoded non-US-ASCII characters. If a [decoding error](http://docs.google.com/nio/charset/CharsetDecoder.html#ce) occurs when decoding the escaped octets then the erroneous octets are replaced by '\uFFFD', the Unicode replacement character.

These operations are exposed in the constructors and methods of this class as follows:

* The [single-argument constructor](http://docs.google.com/java/net/URI.html#URI(java.lang.String)) requires any illegal characters in its argument to be quoted and preserves any escaped octets and *other* characters that are present.
* The [multi-argument constructors](http://docs.google.com/java/net/URI.html#URI(java.lang.String,%20java.lang.String,%20java.lang.String,%20int,%20java.lang.String,%20java.lang.String,%20java.lang.String)) quote illegal characters as required by the components in which they appear. The percent character ('%') is always quoted by these constructors. Any *other* characters are preserved.
* The [getRawUserInfo](http://docs.google.com/java/net/URI.html#getRawUserInfo()), [getRawPath](http://docs.google.com/java/net/URI.html#getRawPath()), [getRawQuery](http://docs.google.com/java/net/URI.html#getRawQuery()), [getRawFragment](http://docs.google.com/java/net/URI.html#getRawFragment()), [getRawAuthority](http://docs.google.com/java/net/URI.html#getRawAuthority()), and [getRawSchemeSpecificPart](http://docs.google.com/java/net/URI.html#getRawSchemeSpecificPart()) methods return the values of their corresponding components in raw form, without interpreting any escaped octets. The strings returned by these methods may contain both escaped octets and *other* characters, and will not contain any illegal characters.
* The [getUserInfo](http://docs.google.com/java/net/URI.html#getUserInfo()), [getPath](http://docs.google.com/java/net/URI.html#getPath()), [getQuery](http://docs.google.com/java/net/URI.html#getQuery()), [getFragment](http://docs.google.com/java/net/URI.html#getFragment()), [getAuthority](http://docs.google.com/java/net/URI.html#getAuthority()), and [getSchemeSpecificPart](http://docs.google.com/java/net/URI.html#getSchemeSpecificPart()) methods decode any escaped octets in their corresponding components. The strings returned by these methods may contain both *other* characters and illegal characters, and will not contain any escaped octets.
* The [toString](http://docs.google.com/java/net/URI.html#toString()) method returns a URI string with all necessary quotation but which may contain *other* characters.
* The [toASCIIString](http://docs.google.com/java/net/URI.html#toASCIIString()) method returns a fully quoted and encoded URI string that does not contain any *other* characters.

#### Identities

For any URI *u*, it is always the case thatnew URI(*u*.toString()).equals(*u*) .For any URI *u* that does not contain redundant syntax such as two slashes before an empty authority (as in file:///tmp/ ) or a colon following a host name but no port (as in http://java.sun.com: ), and that does not encode characters except those that must be quoted, the following identities also hold:new URI(*u*.getScheme(),

*u*.getSchemeSpecificPart(),

*u*.getFragment())

.equals(*u*)in all cases,new URI(*u*.getScheme(),

*u*.getUserInfo(), *u*.getAuthority(),

*u*.getPath(), *u*.getQuery(),

*u*.getFragment())

.equals(*u*)if *u* is hierarchical, andnew URI(*u*.getScheme(),

*u*.getUserInfo(), *u*.getHost(), *u*.getPort(),

*u*.getPath(), *u*.getQuery(),

*u*.getFragment())

.equals(*u*)if *u* is hierarchical and has either no authority or a server-based authority.

#### URIs, URLs, and URNs

A URI is a uniform resource *identifier* while a URL is a uniform resource *locator*. Hence every URL is a URI, abstractly speaking, but not every URI is a URL. This is because there is another subcategory of URIs, uniform resource *names* (URNs), which name resources but do not specify how to locate them. The mailto, news, and isbn URIs shown above are examples of URNs.

The conceptual distinction between URIs and URLs is reflected in the differences between this class and the [URL](http://docs.google.com/java/net/URL.html) class.

An instance of this class represents a URI reference in the syntactic sense defined by RFC 2396. A URI may be either absolute or relative. A URI string is parsed according to the generic syntax without regard to the scheme, if any, that it specifies. No lookup of the host, if any, is performed, and no scheme-dependent stream handler is constructed. Equality, hashing, and comparison are defined strictly in terms of the character content of the instance. In other words, a URI instance is little more than a structured string that supports the syntactic, scheme-independent operations of comparison, normalization, resolution, and relativization.

An instance of the [URL](http://docs.google.com/java/net/URL.html) class, by contrast, represents the syntactic components of a URL together with some of the information required to access the resource that it describes. A URL must be absolute, that is, it must always specify a scheme. A URL string is parsed according to its scheme. A stream handler is always established for a URL, and in fact it is impossible to create a URL instance for a scheme for which no handler is available. Equality and hashing depend upon both the scheme and the Internet address of the host, if any; comparison is not defined. In other words, a URL is a structured string that supports the syntactic operation of resolution as well as the network I/O operations of looking up the host and opening a connection to the specified resource.

**Since:** 1.4 **See Also:**[*RFC 2279: UTF-8, a transformation format of ISO 10646*](http://ietf.org/rfc/rfc2279.txt),

[*RFC 2373: IPv6 Addressing Architecture*](http://www.ietf.org/rfc/rfc2373.txt),

[*RFC 2396: Uniform Resource Identifiers (URI): Generic Syntax*](http://www.ietf.org/rfc/rfc2396.txt),

[*RFC 2732: Format for Literal IPv6 Addresses in URLs*](http://www.ietf.org/rfc/rfc2732.txt),

[URISyntaxException](http://docs.google.com/URISyntaxException.html), [Serialized Form](http://docs.google.com/serialized-form.html#java.net.URI)

| **Constructor Summary** | |
| --- | --- |
| [**URI**](http://docs.google.com/java/net/URI.html#URI(java.lang.String))([String](http://docs.google.com/java/lang/String.html) str)            Constructs a URI by parsing the given string. |
| [**URI**](http://docs.google.com/java/net/URI.html#URI(java.lang.String,%20java.lang.String,%20java.lang.String))([String](http://docs.google.com/java/lang/String.html) scheme, [String](http://docs.google.com/java/lang/String.html) ssp, [String](http://docs.google.com/java/lang/String.html) fragment)            Constructs a URI from the given components. |
| [**URI**](http://docs.google.com/java/net/URI.html#URI(java.lang.String,%20java.lang.String,%20java.lang.String,%20int,%20java.lang.String,%20java.lang.String,%20java.lang.String))([String](http://docs.google.com/java/lang/String.html) scheme, [String](http://docs.google.com/java/lang/String.html) userInfo, [String](http://docs.google.com/java/lang/String.html) host, int port, [String](http://docs.google.com/java/lang/String.html) path, [String](http://docs.google.com/java/lang/String.html) query, [String](http://docs.google.com/java/lang/String.html) fragment)            Constructs a hierarchical URI from the given components. |
| [**URI**](http://docs.google.com/java/net/URI.html#URI(java.lang.String,%20java.lang.String,%20java.lang.String,%20java.lang.String))([String](http://docs.google.com/java/lang/String.html) scheme, [String](http://docs.google.com/java/lang/String.html) host, [String](http://docs.google.com/java/lang/String.html) path, [String](http://docs.google.com/java/lang/String.html) fragment)            Constructs a hierarchical URI from the given components. |
| [**URI**](http://docs.google.com/java/net/URI.html#URI(java.lang.String,%20java.lang.String,%20java.lang.String,%20java.lang.String,%20java.lang.String))([String](http://docs.google.com/java/lang/String.html) scheme, [String](http://docs.google.com/java/lang/String.html) authority, [String](http://docs.google.com/java/lang/String.html) path, [String](http://docs.google.com/java/lang/String.html) query, [String](http://docs.google.com/java/lang/String.html) fragment)            Constructs a hierarchical URI from the given components. |

| **Method Summary** | |
| --- | --- |
| int | [**compareTo**](http://docs.google.com/java/net/URI.html#compareTo(java.net.URI))([URI](http://docs.google.com/java/net/URI.html) that)            Compares this URI to another object, which must be a URI. |
| static [URI](http://docs.google.com/java/net/URI.html) | [**create**](http://docs.google.com/java/net/URI.html#create(java.lang.String))([String](http://docs.google.com/java/lang/String.html) str)            Creates a URI by parsing the given string. |
| boolean | [**equals**](http://docs.google.com/java/net/URI.html#equals(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) ob)            Tests this URI for equality with another object. |
| [String](http://docs.google.com/java/lang/String.html) | [**getAuthority**](http://docs.google.com/java/net/URI.html#getAuthority())()            Returns the decoded authority component of this URI. |
| [String](http://docs.google.com/java/lang/String.html) | [**getFragment**](http://docs.google.com/java/net/URI.html#getFragment())()            Returns the decoded fragment component of this URI. |
| [String](http://docs.google.com/java/lang/String.html) | [**getHost**](http://docs.google.com/java/net/URI.html#getHost())()            Returns the host component of this URI. |
| [String](http://docs.google.com/java/lang/String.html) | [**getPath**](http://docs.google.com/java/net/URI.html#getPath())()            Returns the decoded path component of this URI. |
| int | [**getPort**](http://docs.google.com/java/net/URI.html#getPort())()            Returns the port number of this URI. |
| [String](http://docs.google.com/java/lang/String.html) | [**getQuery**](http://docs.google.com/java/net/URI.html#getQuery())()            Returns the decoded query component of this URI. |
| [String](http://docs.google.com/java/lang/String.html) | [**getRawAuthority**](http://docs.google.com/java/net/URI.html#getRawAuthority())()            Returns the raw authority component of this URI. |
| [String](http://docs.google.com/java/lang/String.html) | [**getRawFragment**](http://docs.google.com/java/net/URI.html#getRawFragment())()            Returns the raw fragment component of this URI. |
| [String](http://docs.google.com/java/lang/String.html) | [**getRawPath**](http://docs.google.com/java/net/URI.html#getRawPath())()            Returns the raw path component of this URI. |
| [String](http://docs.google.com/java/lang/String.html) | [**getRawQuery**](http://docs.google.com/java/net/URI.html#getRawQuery())()            Returns the raw query component of this URI. |
| [String](http://docs.google.com/java/lang/String.html) | [**getRawSchemeSpecificPart**](http://docs.google.com/java/net/URI.html#getRawSchemeSpecificPart())()            Returns the raw scheme-specific part of this URI. |
| [String](http://docs.google.com/java/lang/String.html) | [**getRawUserInfo**](http://docs.google.com/java/net/URI.html#getRawUserInfo())()            Returns the raw user-information component of this URI. |
| [String](http://docs.google.com/java/lang/String.html) | [**getScheme**](http://docs.google.com/java/net/URI.html#getScheme())()            Returns the scheme component of this URI. |
| [String](http://docs.google.com/java/lang/String.html) | [**getSchemeSpecificPart**](http://docs.google.com/java/net/URI.html#getSchemeSpecificPart())()            Returns the decoded scheme-specific part of this URI. |
| [String](http://docs.google.com/java/lang/String.html) | [**getUserInfo**](http://docs.google.com/java/net/URI.html#getUserInfo())()            Returns the decoded user-information component of this URI. |
| int | [**hashCode**](http://docs.google.com/java/net/URI.html#hashCode())()            Returns a hash-code value for this URI. |
| boolean | [**isAbsolute**](http://docs.google.com/java/net/URI.html#isAbsolute())()            Tells whether or not this URI is absolute. |
| boolean | [**isOpaque**](http://docs.google.com/java/net/URI.html#isOpaque())()            Tells whether or not this URI is opaque. |
| [URI](http://docs.google.com/java/net/URI.html) | [**normalize**](http://docs.google.com/java/net/URI.html#normalize())()            Normalizes this URI's path. |
| [URI](http://docs.google.com/java/net/URI.html) | [**parseServerAuthority**](http://docs.google.com/java/net/URI.html#parseServerAuthority())()            Attempts to parse this URI's authority component, if defined, into user-information, host, and port components. |
| [URI](http://docs.google.com/java/net/URI.html) | [**relativize**](http://docs.google.com/java/net/URI.html#relativize(java.net.URI))([URI](http://docs.google.com/java/net/URI.html) uri)            Relativizes the given URI against this URI. |
| [URI](http://docs.google.com/java/net/URI.html) | [**resolve**](http://docs.google.com/java/net/URI.html#resolve(java.lang.String))([String](http://docs.google.com/java/lang/String.html) str)            Constructs a new URI by parsing the given string and then resolving it against this URI. |
| [URI](http://docs.google.com/java/net/URI.html) | [**resolve**](http://docs.google.com/java/net/URI.html#resolve(java.net.URI))([URI](http://docs.google.com/java/net/URI.html) uri)            Resolves the given URI against this URI. |
| [String](http://docs.google.com/java/lang/String.html) | [**toASCIIString**](http://docs.google.com/java/net/URI.html#toASCIIString())()            Returns the content of this URI as a US-ASCII string. |
| [String](http://docs.google.com/java/lang/String.html) | [**toString**](http://docs.google.com/java/net/URI.html#toString())()            Returns the content of this URI as a string. |
| [URL](http://docs.google.com/java/net/URL.html) | [**toURL**](http://docs.google.com/java/net/URI.html#toURL())()            Constructs a URL from this URI. |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [clone](http://docs.google.com/java/lang/Object.html#clone()), [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [notify](http://docs.google.com/java/lang/Object.html#notify()), [notifyAll](http://docs.google.com/java/lang/Object.html#notifyAll()), [wait](http://docs.google.com/java/lang/Object.html#wait()), [wait](http://docs.google.com/java/lang/Object.html#wait(long)), [wait](http://docs.google.com/java/lang/Object.html#wait(long,%20int)) |

| **Constructor Detail** |
| --- |

### URI

public **URI**([String](http://docs.google.com/java/lang/String.html) str)  
 throws [URISyntaxException](http://docs.google.com/java/net/URISyntaxException.html)

Constructs a URI by parsing the given string.

This constructor parses the given string exactly as specified by the grammar in [RFC 2396](http://www.ietf.org/rfc/rfc2396.txt), Appendix A, ***except for the following deviations:***

* An empty authority component is permitted as long as it is followed by a non-empty path, a query component, or a fragment component. This allows the parsing of URIs such as "file:///foo/bar", which seems to be the intent of RFC 2396 although the grammar does not permit it. If the authority component is empty then the user-information, host, and port components are undefined.
* Empty relative paths are permitted; this seems to be the intent of RFC 2396 although the grammar does not permit it. The primary consequence of this deviation is that a standalone fragment such as "#foo" parses as a relative URI with an empty path and the given fragment, and can be usefully [resolved](#4i7ojhp) against a base URI.
* IPv4 addresses in host components are parsed rigorously, as specified by [RFC 2732](http://www.ietf.org/rfc/rfc2732.txt): Each element of a dotted-quad address must contain no more than three decimal digits. Each element is further constrained to have a value no greater than 255.
* Hostnames in host components that comprise only a single domain label are permitted to start with an *alphanum* character. This seems to be the intent of [RFC 2396](http://www.ietf.org/rfc/rfc2396.txt) section 3.2.2 although the grammar does not permit it. The consequence of this deviation is that the authority component of a hierarchical URI such as s://123, will parse as a server-based authority.
* IPv6 addresses are permitted for the host component. An IPv6 address must be enclosed in square brackets ('[' and ']') as specified by [RFC 2732](http://www.ietf.org/rfc/rfc2732.txt). The IPv6 address itself must parse according to [RFC 2373](http://www.ietf.org/rfc/rfc2373.txt). IPv6 addresses are further constrained to describe no more than sixteen bytes of address information, a constraint implicit in RFC 2373 but not expressible in the grammar.
* Characters in the *other* category are permitted wherever RFC 2396 permits *escaped* octets, that is, in the user-information, path, query, and fragment components, as well as in the authority component if the authority is registry-based. This allows URIs to contain Unicode characters beyond those in the US-ASCII character set.

**Parameters:**str - The string to be parsed into a URI **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - If str is null [URISyntaxException](http://docs.google.com/java/net/URISyntaxException.html) - If the given string violates RFC 2396, as augmented by the above deviations

### URI

public **URI**([String](http://docs.google.com/java/lang/String.html) scheme,  
 [String](http://docs.google.com/java/lang/String.html) userInfo,  
 [String](http://docs.google.com/java/lang/String.html) host,  
 int port,  
 [String](http://docs.google.com/java/lang/String.html) path,  
 [String](http://docs.google.com/java/lang/String.html) query,  
 [String](http://docs.google.com/java/lang/String.html) fragment)  
 throws [URISyntaxException](http://docs.google.com/java/net/URISyntaxException.html)

Constructs a hierarchical URI from the given components.

If a scheme is given then the path, if also given, must either be empty or begin with a slash character ('/'). Otherwise a component of the new URI may be left undefined by passing null for the corresponding parameter or, in the case of the port parameter, by passing -1.

This constructor first builds a URI string from the given components according to the rules specified in [RFC 2396](http://www.ietf.org/rfc/rfc2396.txt), section 5.2, step 7:

1. Initially, the result string is empty.
2. If a scheme is given then it is appended to the result, followed by a colon character (':').
3. If user information, a host, or a port are given then the string "//" is appended.
4. If user information is given then it is appended, followed by a commercial-at character ('@'). Any character not in the *unreserved*, *punct*, *escaped*, or *other* categories is [quoted](#tyjcwt).
5. If a host is given then it is appended. If the host is a literal IPv6 address but is not enclosed in square brackets ('[' and ']') then the square brackets are added.
6. If a port number is given then a colon character (':') is appended, followed by the port number in decimal.
7. If a path is given then it is appended. Any character not in the *unreserved*, *punct*, *escaped*, or *other* categories, and not equal to the slash character ('/') or the commercial-at character ('@'), is quoted.
8. If a query is given then a question-mark character ('?') is appended, followed by the query. Any character that is not a [legal URI character](#3znysh7) is quoted.
9. Finally, if a fragment is given then a hash character ('#') is appended, followed by the fragment. Any character that is not a legal URI character is quoted.

The resulting URI string is then parsed as if by invoking the [URI(String)](http://docs.google.com/java/net/URI.html#URI(java.lang.String)) constructor and then invoking the [parseServerAuthority()](http://docs.google.com/java/net/URI.html#parseServerAuthority()) method upon the result; this may cause a [URISyntaxException](http://docs.google.com/java/net/URISyntaxException.html) to be thrown.

**Parameters:**scheme - Scheme nameuserInfo - User name and authorization informationhost - Host nameport - Port numberpath - Pathquery - Queryfragment - Fragment **Throws:** [URISyntaxException](http://docs.google.com/java/net/URISyntaxException.html) - If both a scheme and a path are given but the path is relative, if the URI string constructed from the given components violates RFC 2396, or if the authority component of the string is present but cannot be parsed as a server-based authority

### URI

public **URI**([String](http://docs.google.com/java/lang/String.html) scheme,  
 [String](http://docs.google.com/java/lang/String.html) authority,  
 [String](http://docs.google.com/java/lang/String.html) path,  
 [String](http://docs.google.com/java/lang/String.html) query,  
 [String](http://docs.google.com/java/lang/String.html) fragment)  
 throws [URISyntaxException](http://docs.google.com/java/net/URISyntaxException.html)

Constructs a hierarchical URI from the given components.

If a scheme is given then the path, if also given, must either be empty or begin with a slash character ('/'). Otherwise a component of the new URI may be left undefined by passing null for the corresponding parameter.

This constructor first builds a URI string from the given components according to the rules specified in [RFC 2396](http://www.ietf.org/rfc/rfc2396.txt), section 5.2, step 7:

1. Initially, the result string is empty.
2. If a scheme is given then it is appended to the result, followed by a colon character (':').
3. If an authority is given then the string "//" is appended, followed by the authority. If the authority contains a literal IPv6 address then the address must be enclosed in square brackets ('[' and ']'). Any character not in the *unreserved*, *punct*, *escaped*, or *other* categories, and not equal to the commercial-at character ('@'), is [quoted](#tyjcwt).
4. If a path is given then it is appended. Any character not in the *unreserved*, *punct*, *escaped*, or *other* categories, and not equal to the slash character ('/') or the commercial-at character ('@'), is quoted.
5. If a query is given then a question-mark character ('?') is appended, followed by the query. Any character that is not a [legal URI character](#3znysh7) is quoted.
6. Finally, if a fragment is given then a hash character ('#') is appended, followed by the fragment. Any character that is not a legal URI character is quoted.

The resulting URI string is then parsed as if by invoking the [URI(String)](http://docs.google.com/java/net/URI.html#URI(java.lang.String)) constructor and then invoking the [parseServerAuthority()](http://docs.google.com/java/net/URI.html#parseServerAuthority()) method upon the result; this may cause a [URISyntaxException](http://docs.google.com/java/net/URISyntaxException.html) to be thrown.

**Parameters:**scheme - Scheme nameauthority - Authoritypath - Pathquery - Queryfragment - Fragment **Throws:** [URISyntaxException](http://docs.google.com/java/net/URISyntaxException.html) - If both a scheme and a path are given but the path is relative, if the URI string constructed from the given components violates RFC 2396, or if the authority component of the string is present but cannot be parsed as a server-based authority

### URI

public **URI**([String](http://docs.google.com/java/lang/String.html) scheme,  
 [String](http://docs.google.com/java/lang/String.html) host,  
 [String](http://docs.google.com/java/lang/String.html) path,  
 [String](http://docs.google.com/java/lang/String.html) fragment)  
 throws [URISyntaxException](http://docs.google.com/java/net/URISyntaxException.html)

Constructs a hierarchical URI from the given components.

A component may be left undefined by passing null.

This convenience constructor works as if by invoking the seven-argument constructor as follows:

new [URI](http://docs.google.com/java/net/URI.html#URI(java.lang.String,%20java.lang.String,%20java.lang.String,%20int,%20java.lang.String,%20java.lang.String,%20java.lang.String))(scheme, null, host, -1, path, null, fragment);

**Parameters:**scheme - Scheme namehost - Host namepath - Pathfragment - Fragment **Throws:** [URISyntaxException](http://docs.google.com/java/net/URISyntaxException.html) - If the URI string constructed from the given components violates RFC 2396

### URI

public **URI**([String](http://docs.google.com/java/lang/String.html) scheme,  
 [String](http://docs.google.com/java/lang/String.html) ssp,  
 [String](http://docs.google.com/java/lang/String.html) fragment)  
 throws [URISyntaxException](http://docs.google.com/java/net/URISyntaxException.html)

Constructs a URI from the given components.

A component may be left undefined by passing null.

This constructor first builds a URI in string form using the given components as follows:

1. Initially, the result string is empty.
2. If a scheme is given then it is appended to the result, followed by a colon character (':').
3. If a scheme-specific part is given then it is appended. Any character that is not a [legal URI character](#3znysh7) is [quoted](#tyjcwt).
4. Finally, if a fragment is given then a hash character ('#') is appended to the string, followed by the fragment. Any character that is not a legal URI character is quoted.

The resulting URI string is then parsed in order to create the new URI instance as if by invoking the [URI(String)](http://docs.google.com/java/net/URI.html#URI(java.lang.String)) constructor; this may cause a [URISyntaxException](http://docs.google.com/java/net/URISyntaxException.html) to be thrown.

**Parameters:**scheme - Scheme namessp - Scheme-specific partfragment - Fragment **Throws:** [URISyntaxException](http://docs.google.com/java/net/URISyntaxException.html) - If the URI string constructed from the given components violates RFC 2396

| **Method Detail** |
| --- |

### create

public static [URI](http://docs.google.com/java/net/URI.html) **create**([String](http://docs.google.com/java/lang/String.html) str)

Creates a URI by parsing the given string.

This convenience factory method works as if by invoking the [URI(String)](http://docs.google.com/java/net/URI.html#URI(java.lang.String)) constructor; any [URISyntaxException](http://docs.google.com/java/net/URISyntaxException.html) thrown by the constructor is caught and wrapped in a new [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) object, which is then thrown.

This method is provided for use in situations where it is known that the given string is a legal URI, for example for URI constants declared within in a program, and so it would be considered a programming error for the string not to parse as such. The constructors, which throw [URISyntaxException](http://docs.google.com/java/net/URISyntaxException.html) directly, should be used situations where a URI is being constructed from user input or from some other source that may be prone to errors.

**Parameters:**str - The string to be parsed into a URI **Returns:**The new URI **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - If str is null [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - If the given string violates RFC 2396

### parseServerAuthority

public [URI](http://docs.google.com/java/net/URI.html) **parseServerAuthority**()  
 throws [URISyntaxException](http://docs.google.com/java/net/URISyntaxException.html)

Attempts to parse this URI's authority component, if defined, into user-information, host, and port components.

If this URI's authority component has already been recognized as being server-based then it will already have been parsed into user-information, host, and port components. In this case, or if this URI has no authority component, this method simply returns this URI.

Otherwise this method attempts once more to parse the authority component into user-information, host, and port components, and throws an exception describing why the authority component could not be parsed in that way.

This method is provided because the generic URI syntax specified in [RFC 2396](http://www.ietf.org/rfc/rfc2396.txt) cannot always distinguish a malformed server-based authority from a legitimate registry-based authority. It must therefore treat some instances of the former as instances of the latter. The authority component in the URI string "//foo:bar", for example, is not a legal server-based authority but it is legal as a registry-based authority.

In many common situations, for example when working URIs that are known to be either URNs or URLs, the hierarchical URIs being used will always be server-based. They therefore must either be parsed as such or treated as an error. In these cases a statement such as

URI *u* = new URI(str).parseServerAuthority();

can be used to ensure that *u* always refers to a URI that, if it has an authority component, has a server-based authority with proper user-information, host, and port components. Invoking this method also ensures that if the authority could not be parsed in that way then an appropriate diagnostic message can be issued based upon the exception that is thrown.

**Returns:**A URI whose authority field has been parsed as a server-based authority **Throws:** [URISyntaxException](http://docs.google.com/java/net/URISyntaxException.html) - If the authority component of this URI is defined but cannot be parsed as a server-based authority according to RFC 2396

### normalize

public [URI](http://docs.google.com/java/net/URI.html) **normalize**()

Normalizes this URI's path.

If this URI is opaque, or if its path is already in normal form, then this URI is returned. Otherwise a new URI is constructed that is identical to this URI except that its path is computed by normalizing this URI's path in a manner consistent with [RFC 2396](http://www.ietf.org/rfc/rfc2396.txt), section 5.2, step 6, sub-steps c through f; that is:

1. All "." segments are removed.
2. If a ".." segment is preceded by a non-".." segment then both of these segments are removed. This step is repeated until it is no longer applicable.
3. If the path is relative, and if its first segment contains a colon character (':'), then a "." segment is prepended. This prevents a relative URI with a path such as "a:b/c/d" from later being re-parsed as an opaque URI with a scheme of "a" and a scheme-specific part of "b/c/d". ***(Deviation from RFC 2396)***

A normalized path will begin with one or more ".." segments if there were insufficient non-".." segments preceding them to allow their removal. A normalized path will begin with a "." segment if one was inserted by step 3 above. Otherwise, a normalized path will not contain any "." or ".." segments.

**Returns:**A URI equivalent to this URI, but whose path is in normal form

### resolve

public [URI](http://docs.google.com/java/net/URI.html) **resolve**([URI](http://docs.google.com/java/net/URI.html) uri)

Resolves the given URI against this URI.

If the given URI is already absolute, or if this URI is opaque, then the given URI is returned.

If the given URI's fragment component is defined, its path component is empty, and its scheme, authority, and query components are undefined, then a URI with the given fragment but with all other components equal to those of this URI is returned. This allows a URI representing a standalone fragment reference, such as "#foo", to be usefully resolved against a base URI.

Otherwise this method constructs a new hierarchical URI in a manner consistent with [RFC 2396](http://www.ietf.org/rfc/rfc2396.txt), section 5.2; that is:

1. A new URI is constructed with this URI's scheme and the given URI's query and fragment components.
2. If the given URI has an authority component then the new URI's authority and path are taken from the given URI.
3. Otherwise the new URI's authority component is copied from this URI, and its path is computed as follows:
   1. If the given URI's path is absolute then the new URI's path is taken from the given URI.
   2. Otherwise the given URI's path is relative, and so the new URI's path is computed by resolving the path of the given URI against the path of this URI. This is done by concatenating all but the last segment of this URI's path, if any, with the given URI's path and then normalizing the result as if by invoking the [normalize](http://docs.google.com/java/net/URI.html#normalize()) method.

The result of this method is absolute if, and only if, either this URI is absolute or the given URI is absolute.

**Parameters:**uri - The URI to be resolved against this URI **Returns:**The resulting URI **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - If uri is null

### resolve

public [URI](http://docs.google.com/java/net/URI.html) **resolve**([String](http://docs.google.com/java/lang/String.html) str)

Constructs a new URI by parsing the given string and then resolving it against this URI.

This convenience method works as if invoking it were equivalent to evaluating the expression [resolve](http://docs.google.com/java/net/URI.html#resolve(java.net.URI))(URI.[create](http://docs.google.com/java/net/URI.html#create(java.lang.String))(str)).

**Parameters:**str - The string to be parsed into a URI **Returns:**The resulting URI **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - If str is null [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - If the given string violates RFC 2396

### relativize

public [URI](http://docs.google.com/java/net/URI.html) **relativize**([URI](http://docs.google.com/java/net/URI.html) uri)

Relativizes the given URI against this URI.

The relativization of the given URI against this URI is computed as follows:

1. If either this URI or the given URI are opaque, or if the scheme and authority components of the two URIs are not identical, or if the path of this URI is not a prefix of the path of the given URI, then the given URI is returned.
2. Otherwise a new relative hierarchical URI is constructed with query and fragment components taken from the given URI and with a path component computed by removing this URI's path from the beginning of the given URI's path.

**Parameters:**uri - The URI to be relativized against this URI **Returns:**The resulting URI **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - If uri is null

### toURL

public [URL](http://docs.google.com/java/net/URL.html) **toURL**()  
 throws [MalformedURLException](http://docs.google.com/java/net/MalformedURLException.html)

Constructs a URL from this URI.

This convenience method works as if invoking it were equivalent to evaluating the expression new URL(this.toString()) after first checking that this URI is absolute.

**Returns:**A URL constructed from this URI **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - If this URL is not absolute [MalformedURLException](http://docs.google.com/java/net/MalformedURLException.html) - If a protocol handler for the URL could not be found, or if some other error occurred while constructing the URL

### getScheme

public [String](http://docs.google.com/java/lang/String.html) **getScheme**()

Returns the scheme component of this URI.

The scheme component of a URI, if defined, only contains characters in the *alphanum* category and in the string "-.+". A scheme always starts with an *alpha* character.

The scheme component of a URI cannot contain escaped octets, hence this method does not perform any decoding.

**Returns:**The scheme component of this URI, or null if the scheme is undefined

### isAbsolute

public boolean **isAbsolute**()

Tells whether or not this URI is absolute.

A URI is absolute if, and only if, it has a scheme component.

**Returns:**true if, and only if, this URI is absolute

### isOpaque

public boolean **isOpaque**()

Tells whether or not this URI is opaque.

A URI is opaque if, and only if, it is absolute and its scheme-specific part does not begin with a slash character ('/'). An opaque URI has a scheme, a scheme-specific part, and possibly a fragment; all other components are undefined.

**Returns:**true if, and only if, this URI is opaque

### getRawSchemeSpecificPart

public [String](http://docs.google.com/java/lang/String.html) **getRawSchemeSpecificPart**()

Returns the raw scheme-specific part of this URI. The scheme-specific part is never undefined, though it may be empty.

The scheme-specific part of a URI only contains legal URI characters.

**Returns:**The raw scheme-specific part of this URI (never null)

### getSchemeSpecificPart

public [String](http://docs.google.com/java/lang/String.html) **getSchemeSpecificPart**()

Returns the decoded scheme-specific part of this URI.

The string returned by this method is equal to that returned by the [getRawSchemeSpecificPart](http://docs.google.com/java/net/URI.html#getRawSchemeSpecificPart()) method except that all sequences of escaped octets are [decoded](#3dy6vkm).

**Returns:**The decoded scheme-specific part of this URI (never null)

### getRawAuthority

public [String](http://docs.google.com/java/lang/String.html) **getRawAuthority**()

Returns the raw authority component of this URI.

The authority component of a URI, if defined, only contains the commercial-at character ('@') and characters in the *unreserved*, *punct*, *escaped*, and *other* categories. If the authority is server-based then it is further constrained to have valid user-information, host, and port components.

**Returns:**The raw authority component of this URI, or null if the authority is undefined

### getAuthority

public [String](http://docs.google.com/java/lang/String.html) **getAuthority**()

Returns the decoded authority component of this URI.

The string returned by this method is equal to that returned by the [getRawAuthority](http://docs.google.com/java/net/URI.html#getRawAuthority()) method except that all sequences of escaped octets are [decoded](#3dy6vkm).

**Returns:**The decoded authority component of this URI, or null if the authority is undefined

### getRawUserInfo

public [String](http://docs.google.com/java/lang/String.html) **getRawUserInfo**()

Returns the raw user-information component of this URI.

The user-information component of a URI, if defined, only contains characters in the *unreserved*, *punct*, *escaped*, and *other* categories.

**Returns:**The raw user-information component of this URI, or null if the user information is undefined

### getUserInfo

public [String](http://docs.google.com/java/lang/String.html) **getUserInfo**()

Returns the decoded user-information component of this URI.

The string returned by this method is equal to that returned by the [getRawUserInfo](http://docs.google.com/java/net/URI.html#getRawUserInfo()) method except that all sequences of escaped octets are [decoded](#3dy6vkm).

**Returns:**The decoded user-information component of this URI, or null if the user information is undefined

### getHost

public [String](http://docs.google.com/java/lang/String.html) **getHost**()

Returns the host component of this URI.

The host component of a URI, if defined, will have one of the following forms:

* A domain name consisting of one or more *labels* separated by period characters ('.'), optionally followed by a period character. Each label consists of *alphanum* characters as well as hyphen characters ('-'), though hyphens never occur as the first or last characters in a label. The rightmost label of a domain name consisting of two or more labels, begins with an *alpha* character.
* A dotted-quad IPv4 address of the form *digit*+.*digit*+.*digit*+.*digit*+, where no *digit* sequence is longer than three characters and no sequence has a value larger than 255.
* An IPv6 address enclosed in square brackets ('[' and ']') and consisting of hexadecimal digits, colon characters (':'), and possibly an embedded IPv4 address. The full syntax of IPv6 addresses is specified in [*RFC 2373: IPv6 Addressing Architecture*](http://www.ietf.org/rfc/rfc2373.txt).

The host component of a URI cannot contain escaped octets, hence this method does not perform any decoding.

**Returns:**The host component of this URI, or null if the host is undefined

### getPort

public int **getPort**()

Returns the port number of this URI.

The port component of a URI, if defined, is a non-negative integer.

**Returns:**The port component of this URI, or -1 if the port is undefined

### getRawPath

public [String](http://docs.google.com/java/lang/String.html) **getRawPath**()

Returns the raw path component of this URI.

The path component of a URI, if defined, only contains the slash character ('/'), the commercial-at character ('@'), and characters in the *unreserved*, *punct*, *escaped*, and *other* categories.

**Returns:**The path component of this URI, or null if the path is undefined

### getPath

public [String](http://docs.google.com/java/lang/String.html) **getPath**()

Returns the decoded path component of this URI.

The string returned by this method is equal to that returned by the [getRawPath](http://docs.google.com/java/net/URI.html#getRawPath()) method except that all sequences of escaped octets are [decoded](#3dy6vkm).

**Returns:**The decoded path component of this URI, or null if the path is undefined

### getRawQuery

public [String](http://docs.google.com/java/lang/String.html) **getRawQuery**()

Returns the raw query component of this URI.

The query component of a URI, if defined, only contains legal URI characters.

**Returns:**The raw query component of this URI, or null if the query is undefined

### getQuery

public [String](http://docs.google.com/java/lang/String.html) **getQuery**()

Returns the decoded query component of this URI.

The string returned by this method is equal to that returned by the [getRawQuery](http://docs.google.com/java/net/URI.html#getRawQuery()) method except that all sequences of escaped octets are [decoded](#3dy6vkm).

**Returns:**The decoded query component of this URI, or null if the query is undefined

### getRawFragment

public [String](http://docs.google.com/java/lang/String.html) **getRawFragment**()

Returns the raw fragment component of this URI.

The fragment component of a URI, if defined, only contains legal URI characters.

**Returns:**The raw fragment component of this URI, or null if the fragment is undefined

### getFragment

public [String](http://docs.google.com/java/lang/String.html) **getFragment**()

Returns the decoded fragment component of this URI.

The string returned by this method is equal to that returned by the [getRawFragment](http://docs.google.com/java/net/URI.html#getRawFragment()) method except that all sequences of escaped octets are [decoded](#3dy6vkm).

**Returns:**The decoded fragment component of this URI, or null if the fragment is undefined

### equals

public boolean **equals**([Object](http://docs.google.com/java/lang/Object.html) ob)

Tests this URI for equality with another object.

If the given object is not a URI then this method immediately returns false.

For two URIs to be considered equal requires that either both are opaque or both are hierarchical. Their schemes must either both be undefined or else be equal without regard to case. Their fragments must either both be undefined or else be equal.

For two opaque URIs to be considered equal, their scheme-specific parts must be equal.

For two hierarchical URIs to be considered equal, their paths must be equal and their queries must either both be undefined or else be equal. Their authorities must either both be undefined, or both be registry-based, or both be server-based. If their authorities are defined and are registry-based, then they must be equal. If their authorities are defined and are server-based, then their hosts must be equal without regard to case, their port numbers must be equal, and their user-information components must be equal.

When testing the user-information, path, query, fragment, authority, or scheme-specific parts of two URIs for equality, the raw forms rather than the encoded forms of these components are compared and the hexadecimal digits of escaped octets are compared without regard to case.

This method satisfies the general contract of the [Object.equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)) method.

**Overrides:**[equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)) in class [Object](http://docs.google.com/java/lang/Object.html) **Parameters:**ob - The object to which this object is to be compared **Returns:**true if, and only if, the given object is a URI that is identical to this URI**See Also:**[Object.hashCode()](http://docs.google.com/java/lang/Object.html#hashCode()), [Hashtable](http://docs.google.com/java/util/Hashtable.html)

### hashCode

public int **hashCode**()

Returns a hash-code value for this URI. The hash code is based upon all of the URI's components, and satisfies the general contract of the [Object.hashCode](http://docs.google.com/java/lang/Object.html#hashCode()) method.

**Overrides:**[hashCode](http://docs.google.com/java/lang/Object.html#hashCode()) in class [Object](http://docs.google.com/java/lang/Object.html) **Returns:**A hash-code value for this URI**See Also:**[Object.equals(java.lang.Object)](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [Hashtable](http://docs.google.com/java/util/Hashtable.html)

### compareTo

public int **compareTo**([URI](http://docs.google.com/java/net/URI.html) that)

Compares this URI to another object, which must be a URI.

When comparing corresponding components of two URIs, if one component is undefined but the other is defined then the first is considered to be less than the second. Unless otherwise noted, string components are ordered according to their natural, case-sensitive ordering as defined by the [String.compareTo](http://docs.google.com/java/lang/Comparable.html#compareTo(T)) method. String components that are subject to encoding are compared by comparing their raw forms rather than their encoded forms.

The ordering of URIs is defined as follows:

* Two URIs with different schemes are ordered according the ordering of their schemes, without regard to case.
* A hierarchical URI is considered to be less than an opaque URI with an identical scheme.
* Two opaque URIs with identical schemes are ordered according to the ordering of their scheme-specific parts.
* Two opaque URIs with identical schemes and scheme-specific parts are ordered according to the ordering of their fragments.
* Two hierarchical URIs with identical schemes are ordered according to the ordering of their authority components:
  + If both authority components are server-based then the URIs are ordered according to their user-information components; if these components are identical then the URIs are ordered according to the ordering of their hosts, without regard to case; if the hosts are identical then the URIs are ordered according to the ordering of their ports.
  + If one or both authority components are registry-based then the URIs are ordered according to the ordering of their authority components.
* Finally, two hierarchical URIs with identical schemes and authority components are ordered according to the ordering of their paths; if their paths are identical then they are ordered according to the ordering of their queries; if the queries are identical then they are ordered according to the order of their fragments.

This method satisfies the general contract of the [Comparable.compareTo](http://docs.google.com/java/lang/Comparable.html#compareTo(T)) method.

**Specified by:**[compareTo](http://docs.google.com/java/lang/Comparable.html#compareTo(T)) in interface [Comparable](http://docs.google.com/java/lang/Comparable.html)<[URI](http://docs.google.com/java/net/URI.html)> **Parameters:**that - The object to which this URI is to be compared **Returns:**A negative integer, zero, or a positive integer as this URI is less than, equal to, or greater than the given URI **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - If the given object is not a URI

### toString

public [String](http://docs.google.com/java/lang/String.html) **toString**()

Returns the content of this URI as a string.

If this URI was created by invoking one of the constructors in this class then a string equivalent to the original input string, or to the string computed from the originally-given components, as appropriate, is returned. Otherwise this URI was created by normalization, resolution, or relativization, and so a string is constructed from this URI's components according to the rules specified in [RFC 2396](http://www.ietf.org/rfc/rfc2396.txt), section 5.2, step 7.

**Overrides:**[toString](http://docs.google.com/java/lang/Object.html#toString()) in class [Object](http://docs.google.com/java/lang/Object.html) **Returns:**The string form of this URI

### toASCIIString

public [String](http://docs.google.com/java/lang/String.html) **toASCIIString**()

Returns the content of this URI as a US-ASCII string.

If this URI does not contain any characters in the *other* category then an invocation of this method will return the same value as an invocation of the [toString](http://docs.google.com/java/net/URI.html#toString()) method. Otherwise this method works as if by invoking that method and then [encoding](#2et92p0) the result.

**Returns:**The string form of this URI, encoded as needed so that it only contains characters in the US-ASCII charset

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/URI.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/net/UnknownServiceException.html)   [**NEXT CLASS**](http://docs.google.com/java/net/URISyntaxException.html) | [**FRAMES**](http://docs.google.com/index.html?java/net/URI.html)    [**NO FRAMES**](http://docs.google.com/URI.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#1t3h5sf) | [METHOD](#4d34og8) | DETAIL: FIELD | [CONSTR](#17dp8vu) | [METHOD](#44sinio) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).