InetAddress、Inet4Address、Inet6Address

**Inet4Address** This class represents an Internet Protocol version 4 (IPv4) address.

**Inet6Address** This class represents an Internet Protocol version 6 (IPv6) address.

**InetAddress** This class represents an Internet Protocol (IP) address.

# InetAddress

## 继承关系

public class **InetAddress** extends Object implements **Serializable**



All Implemented Interfaces: Serializable

Direct Known Subclasses: **Inet4Address**, **Inet6Address**

## 功能：代表一个IP地址。

**This class represents an Internet Protocol (IP) address.**

An IP address is either **a 32-bit or 128-bit unsigned number** used by IP, a lower-level protocol on which protocols like UDP and TCP are built. The IP address architecture is defined *by RFC 790: Assigned Numbers, RFC 1918: Address Allocation for Private Internets, RFC 2365: Administratively Scoped IP Multicast, and RFC 2373: IP Version 6 Addressing Architecture*. An instance of an InetAddress consists of an IP address and possibly its corresponding host name (depending on whether it is constructed with a host name or whether it has already done reverse host name resolution).

InetAddress实例由IP地址和其相关的主机名称组成。

## IP地址类型Address types

### **unicast**

**An identifier for a single interface.** A packet sent to a unicast address is delivered to the interface identified by that address.

**The Unspecified Address** -- Also called anylocal or wildcard address. It must never be assigned to any node. It indicates the absence of an address. One example of its use is as the target of bind, which allows a server to accept a client connection on any interface, in case the server host has multiple interfaces.

The unspecified address must not be used as the destination address of an IP packet.

**The Loopback Addresses** -- This is the address assigned to the loopback interface. Anything sent to this IP address loops around and becomes IP input on the local host. This address is often used when testing a client.

### **multicast**

An identifier for a set of interfaces (typically belonging to different nodes). A packet sent to a multicast address is delivered to all interfaces identified by that address.

## IP address scope

**Link-local addresses** are designed to be used for addressing on a single link for purposes such as auto-address configuration, neighbor discovery, or when no routers are present.

**Site-local addresses** are designed to be used for addressing inside of a site without the need for a global prefix.

**Global addresses** are unique across the internet.

## Textual representation of IP addresses

textual 英 [ˈtekstʃuəl] 美 [ˈtɛkstʃuəl] adj.正文的，版本的，原文的

The textual representation of an IP address is address family specific.

For IPv4 address format, please refer to **Inet4Address#format**; For IPv6 address format, please refer to **Inet6Address#format.**

There is a couple of System Properties affecting how IPv4 and IPv6 addresses are used.

## Host Name Resolution

Host name-to-IP address resolution is accomplished through the use of *a combination of local machine configuration information and network naming services* such as **the Domain Name System (DNS) and Network Information Service(NIS)**. The particular naming services(s) being used is by default the local machine configured one. For any host name, its corresponding IP address is returned.

Reverse name resolution means that for any IP address, the host associated with the IP address is returned.

The **InetAddress** class provides methods to resolve host names to their IP addresses and vice versa.

## InetAddress Caching

The InetAddress class has a cache to store successful as well as unsuccessful host name resolutions.

By default, when a security manager is installed, in order to protect against DNS spoofing attacks, the result of positive host name resolutions are cached forever. When a security manager is not installed, the default behavior is to cache entries for a finite (implementation dependent) period of time. The result of unsuccessful host name resolution is cached for a very short period of time (10 seconds) to improve performance.

If the default behavior is not desired, then **a Java security property** can be set to a different **Time-to-live (TTL)** value for positive caching. Likewise, a system admin can configure a different negative caching TTL value when needed.

Two Java security properties control the TTL values used for positive and negative host name resolution caching:

**networkaddress.cache.ttl**

Indicates the caching policy for successful name lookups from the name service. The value is specified as as integer to indicate the number of seconds to cache the successful lookup. The default setting is to cache for an implementation specific period of time.

A value of -1 indicates "cache forever".

**networkaddress.cache.negative.ttl (default: 10)**

Indicates the caching policy for un-successful name lookups from the name service. The value is specified as as integer to indicate the number of seconds to cache the failure for un-successful lookups.

A value of 0 indicates "never cache". A value of -1 indicates "cache forever".

## 没有构造方法：通过静态静态方法getByAddress或getByName获取InetAddress实例。

## 创建InetAddress实例的静态方法

### static InetAddress **getByAddress(byte[] addr)**

Returns an InetAddress object given the raw IP address .

### static InetAddress **getByAddress(String host, byte[] addr)**

Creates an InetAddress based on the provided host name and IP address.

### static InetAddress **getByName(String host)**

Determines the IP address of a host, given the host's name.

### static InetAddress[] getAllByName(String host)

**Given the name of a host,** returns an array of its IP addresses, based on the configured name service on the system.(一个主机名称对应多个IP地址)

### static InetAddress **getLocalHost()**

Returns the address of the local host.

### static InetAddress **getLoopbackAddress()**

Returns the loopback address.

## 其他一般方法

### getXxx方法

byte[] **getAddress**()：获取原生IP地址(字节数组)

Returns the raw IP address of this InetAddress object.

String **getCanonicalHostName**()

Gets the fully qualified domain name for this IP address.

String **getHostAddress**()

Returns the IP address string in textual presentation.

String **getHostName**()

Gets the host name for this IP address.

### isXxx判断方法

boolean **isAnyLocalAddress**()

Utility routine to check if the InetAddress in a wildcard address.

boolean **isLinkLocalAddress**()

Utility routine to check if the InetAddress is an link local address.

boolean **isLoopbackAddress**()

Utility routine to check if the InetAddress is a loopback address.

boolean **isMCGlobal**()

Utility routine to check if the multicast address has global scope.

boolean **isMCLinkLocal**()

Utility routine to check if the multicast address has link scope.

boolean **isMCNodeLocal**()

Utility routine to check if the multicast address has node scope.

boolean **isMCOrgLocal**()

Utility routine to check if the multicast address has organization scope.

boolean **isMCSiteLocal**()

Utility routine to check if the multicast address has site scope.

boolean **isMulticastAddress**()

Utility routine to check if the InetAddress is an IP multicast address.

boolean **isReachable**(int timeout)

Test whether that address is reachable.

**boolean isReachable(NetworkInterface netif, int ttl, int timeout)**

Test whether that address is reachable.

**boolean isSiteLocalAddress()**

Utility routine to check if the InetAddress is a site local address.

### boolean equals(Object obj)

Compares this object against the specified object.

### String toString()

Converts this IP address to a String.

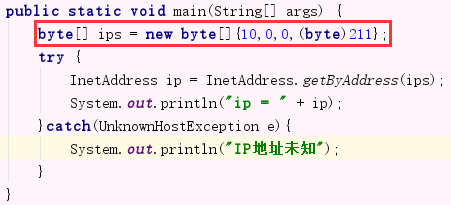
### int hashCode()

Returns a hashcode for this IP address.

# 使用示例

## byte类型数据范围为-128—127，那么128-255如何表示呢？

利用**byte**进行强制类型转换即可。



输出结果为：ip = /10.0.0.211

# Inet4Address

## 继承关系

public final class **Inet4Address** extends **InetAddress**

**存在java.net包中。**



All Implemented Interfaces: Serializable

## 功能特性

This class represents **an Internet Protocol version 4 (IPv4) address**. Defined by RFC 790: Assigned Numbers, RFC 1918: Address Allocation for Private Internets, and RFC 2365: Administratively Scoped IP Multicast

Textual representation of IP addresses

Textual representation of IPv4 address used as input to methods takes one of the following forms:

**d.d.d.d**

**d.d.d**

**d.d**

**d**

When four parts are specified, each is interpreted as a byte of data and assigned, from left to right, to the four bytes of an IPv4 address.

When a three part address is specified, the last part is interpreted as a 16-bit quantity and placed in the right most two bytes of the network address. This makes the three part address format convenient for specifying Class B net- work addresses as 128.net.host.

When a two part address is supplied, the last part is interpreted as a 24-bit quantity and placed in the right most three bytes of the network address. This makes the two part address format convenient for specifying Class A network addresses as **net.host**.

When only one part is given, the value is stored directly in the network address without any byte rearrangement.

For methods that return a textual representation as output value, the first form, i.e. a dotted-quad string, is used.

The Scope of a Multicast Address

Historically the IPv4 TTL field in the IP header has doubled as a multicast scope field: a TTL of 0 means node-local, 1 means link-local, up through 32 means site-local, up through 64 means region-local, up through 128 means continent-local, and up through 255 are global. However, the administrative scoping is preferred. Please refer to RFC 2365: Administratively Scoped IP Multicast

## 方法同InetAddress。

# InetAddress

## 继承关系

public final class **Inet6Address** extends **InetAddress**



All Implemented Interfaces: Serializable

## 功能特性

This class represents **an Internet Protocol version 6 (IPv6) address**. Defined by RFC 2373: IP Version 6 Addressing Architecture.

IPv6的特性见文档介绍。这里不列举了。

## 静态方法

**static Inet6Address getByAddress(String host, byte[] addr, int scope\_id)**

Create an Inet6Address in the exact manner of InetAddress.getByAddress(String,byte[]) except that the IPv6 scope\_id is set to the given numeric value.

**static Inet6Address getByAddress(String host, byte[] addr, NetworkInterface nif)**

Create an Inet6Address in the exact manner of InetAddress.getByAddress(String,byte[]) except that the IPv6 scope\_id is set to the value corresponding to the given interface for the address type specified in addr.

## 一般方法

特有的方法：

NetworkInterface **getScopedInterface**()

Returns the scoped interface, if this instance was created with with a scoped interface.

boolean **isIPv4CompatibleAddress**()是否合IPv4地址兼容

Utility routine to check if the InetAddress is an IPv4 compatible IPv6 address.