compact2, compact3

java.rmi.server

Class UnicastRemoteObject

java.lang.Object
  java.rmi.server.RemoteObject
    java.rmi.server.RemoteServer
      java.rmi.server.UnicastRemoteObject

All Implemented Interfaces:

`Serializable`, `Remote`

Direct Known Subclasses:

`ActivationGroup`

```
public class UnicastRemoteObject
extends RemoteServer
```

Used for exporting a remote object with JRMP and obtaining a stub that communicates to the remote object. Stubs are either generated at runtime using dynamic proxy objects, or they are generated statically at build time, typically using the `rmic` tool.

Deprecated: Static Stubs. *Support for statically generated stubs is deprecated. This includes the API in this class that requires the use of static stubs, as well as the runtime support for loading static stubs. Generating stubs dynamically is preferred, using one of the five non-deprecated ways of exporting objects as listed below. Do not run `rmic` to generate static stub classes. It is unnecessary, and it is also deprecated.*

There are six ways to export remote objects:

1. Subclassing `UnicastRemoteObject` and calling the `UnicastRemoteObject()` constructor.
2. Subclassing `UnicastRemoteObject` and calling the `UnicastRemoteObject(port)` constructor.
3. Subclassing `UnicastRemoteObject` and calling the `UnicastRemoteObject(port, csf, ssf)` constructor.
4. Calling the `exportObject(Remote)` method. **Deprecated.**

5. Calling the `exportObject(Remote, port)` method.
6. Calling the `exportObject(Remote, port, csf, ssf)` method.

The fourth technique, `exportObject(Remote)`, always uses statically generated stubs and is deprecated.

The other five techniques all use the following approach: if the `java.rmi.server.ignoreStubClasses` property is `true` (case insensitive) or if a static stub cannot be found, stubs are generated dynamically using `Proxy` objects. Otherwise, static stubs are used.

The default value of the `java.rmi.server.ignoreStubClasses` property is `false`.

Statically generated stubs are typically pregenerated from the remote object's class using the `rmic` tool. A static stub is loaded and an instance of that stub class is constructed as described below.

- A "root class" is determined as follows: if the remote object's class directly implements an interface that extends `Remote`, then the remote object's class is the root class; otherwise, the root class is the most derived superclass of the remote object's class that directly implements an interface that extends `Remote`.
- The name of the stub class to load is determined by concatenating the binary name of the root class with the suffix `_Stub`.
- The stub class is loaded by name using the class loader of the root class. The stub class must extend `RemoteStub` and must have a public constructor that has one parameter of type `RemoteRef`.
- Finally, an instance of the stub class is constructed with a `RemoteRef`.
- If the appropriate stub class could not be found, or if the stub class could not be loaded, or if a problem occurs creating the stub instance, a `StubNotFoundException` is thrown.

Stubs are dynamically generated by constructing an instance of a `Proxy` with the following characteristics:

- The proxy's class is defined by the class loader of the remote object's class.
- The proxy implements all the remote interfaces implemented by the remote object's class.
- The proxy's invocation handler is a `RemoteObjectInvocationHandler` instance constructed with a `RemoteRef`.
- If the proxy could not be created, a `StubNotFoundException` will be thrown.

Implementation Note:

Depending upon which constructor or static method is used for exporting an object, `RMISocketFactory` may be used for creating sockets. By default, server sockets created by `RMISocketFactory` listen on all network interfaces. See the `RMISocketFactory` class and the section `RMI Socket Factories` in the `Java RMI Specification`.

Since:

JDK1.1

See Also:

`Serialized Form`

Field Summary

Fields inherited from class java.rmi.server.RemoteObject

`ref`

Constructor Summary

Constructors

Modifier	Constructor and Description
protected	**UnicastRemoteObject**() Creates and exports a new UnicastRemoteObject object using an anonymous port.
protected	**UnicastRemoteObject**(int port) Creates and exports a new UnicastRemoteObject object using the particular supplied port.
protected	**UnicastRemoteObject**(int port, RMIClientSocketFactory csf, RMIServerSocketFactory ssf) Creates and exports a new UnicastRemoteObject object using the particular supplied port and socket factories.

Method Summary

All Methods | **Static Methods** | **Instance Methods** | **Concrete Methods** | **Deprecated Methods**

Modifier and Type	Method and Description
Object	**clone**() Returns a clone of the remote object that is distinct from the original.
static RemoteStub	**exportObject**(Remote obj) **Deprecated.**

	This method is deprecated because it supports only static stubs. Use **exportObject(Remote, port)** or **exportObject(Remote, port, csf, ssf)** instead.
static **Remote**	**exportObject**(**Remote** obj, int port) Exports the remote object to make it available to receive incoming calls, using the particular supplied port.
static **Remote**	**exportObject**(**Remote** obj, int port, **RMIClientSocketFactory** csf, **RMIServerSocketFactory** ssf) Exports the remote object to make it available to receive incoming calls, using a transport specified by the given socket factory.
static boolean	**unexportObject**(**Remote** obj, boolean force) Removes the remote object, obj, from the RMI runtime.

Methods inherited from class java.rmi.server.RemoteServer

getClientHost, getLog, setLog

Methods inherited from class java.rmi.server.RemoteObject

equals, getRef, hashCode, toString, toStub

Methods inherited from class java.lang.Object

finalize, getClass, notify, notifyAll, wait, wait, wait

Constructor Detail

UnicastRemoteObject

```
protected UnicastRemoteObject()
                       throws RemoteException
```

Creates and exports a new UnicastRemoteObject object using an anonymous port.

The object is exported with a server socket created using the RMISocketFactory class.

Throws:

RemoteException - if failed to export object

Since:

JDK1.1

UnicastRemoteObject

```
protected UnicastRemoteObject(int port)
                       throws RemoteException
```

Creates and exports a new UnicastRemoteObject object using the particular supplied port.

The object is exported with a server socket created using the RMISocketFactory class.

Parameters:

port - the port number on which the remote object receives calls (if port is zero, an anonymous port is chosen)

Throws:

RemoteException - if failed to export object

Since:

1.2

UnicastRemoteObject

```
protected UnicastRemoteObject(int port,
                              RMIClientSocketFactory csf,
                              RMIServerSocketFactory ssf)
                       throws RemoteException
```

Creates and exports a new UnicastRemoteObject object using the particular supplied port and socket factories.

Either socket factory may be `null`, in which case the corresponding client or server socket creation method of RMISocketFactory is used instead.

Parameters:

port - the port number on which the remote object receives calls (if port is zero, an anonymous port is chosen)

csf - the client-side socket factory for making calls to the remote object

ssf - the server-side socket factory for receiving remote calls

Throws:

RemoteException - if failed to export object

Since:

1.2

Method Detail

clone

```
public Object clone()
             throws CloneNotSupportedException
```

Returns a clone of the remote object that is distinct from the original.

Overrides:

clone in class Object

Returns:

the new remote object

Throws:

CloneNotSupportedException - if clone failed due to a RemoteException.

Since:

JDK1.1

See Also:

Cloneable

exportObject

```
@Deprecated
public static RemoteStub exportObject(Remote obj)
                                       throws RemoteException
```

Deprecated. *This method is deprecated because it supports only static stubs. Use `exportObject(Remote, port)` or `exportObject(Remote, port, csf, ssf)` instead.*

Exports the remote object to make it available to receive incoming calls using an anonymous port. This method will always return a statically generated stub.

The object is exported with a server socket created using the `RMISocketFactory` class.

Parameters:

`obj - the remote object to be exported`

Returns:

`remote object stub`

Throws:

`RemoteException - if export fails`

Since:

`JDK1.1`

exportObject

```
public static Remote exportObject(Remote obj,
                                  int port)
                           throws RemoteException
```

Exports the remote object to make it available to receive incoming calls, using the particular supplied port.

The object is exported with a server socket created using the `RMISocketFactory` class.

Parameters:

`obj - the remote object to be exported`

port - the port to export the object on

Returns:

remote object stub

Throws:

RemoteException - if export fails

Since:

1.2

exportObject

```
public static Remote exportObject(Remote obj,
                                  int port,
                                  RMIClientSocketFactory csf,
                                  RMIServerSocketFactory ssf)
                           throws RemoteException
```

Exports the remote object to make it available to receive incoming calls, using a transport specified by the given socket factory.

Either socket factory may be `null`, in which case the corresponding client or server socket creation method of `RMISocketFactory` is used instead.

Parameters:

obj - the remote object to be exported

port - the port to export the object on

csf - the client-side socket factory for making calls to the remote object

ssf - the server-side socket factory for receiving remote calls

Returns:

remote object stub

Throws:

RemoteException - if export fails

Since:

1.2

unexportObject

```
public static boolean unexportObject(Remote obj,
                                     boolean force)
                              throws NoSuchObjectException
```

Removes the remote object, obj, from the RMI runtime. If successful, the object can no longer accept incoming RMI calls. If the force parameter is true, the object is forcibly unexported even if there are pending calls to the remote object or the remote object still has calls in progress. If the force parameter is false, the object is only unexported if there are no pending or in progress calls to the object.

Parameters:

obj - the remote object to be unexported

force - if true, unexports the object even if there are pending or in-progress calls; if false, only unexports the object if there are no pending or in-progress calls

Returns:

true if operation is successful, false otherwise

Throws:

NoSuchObjectException - if the remote object is not currently exported

Since:

1.2