

# Table of Contents

The server for JVx clients is instantiated either at the current JVM or at the application server (e.g., Tomcat, JBoss). In the current VM, the call

```
Server server = new Server();
```

is sufficient to create a server instance. However, this creates that risk that multiple server instances are created (which may well be desirable). The following method is available to treat the server as a singleton:

```
Server server = Server.getInstance();
```

In doing so, we additionally attempt to address a server instance via JNDI.

A new instance of the server is created at the application server using ServletServer. This is accomplished by the following call:

```
Server server = Server.getInstance();
```

If the server is available as a JNDI resource, the application server handles the instantiation of the server. This way one server instance could be used for all applications of an application server.

A global JNDI resource for the application server Tomcat is configured as follows:

**conf/server.xml:**

[server.xml](#)

```
<GlobalNamingResources>
    ...
    ...
    ...
    <Resource auth="Container"
               factory="org.apache.naming.factory.BeanFactory"
               name="globalserver"
               type="com.sibvisions.rad.server.Server"/>
</GlobalNamingResources>

<DefaultContext>
    <ResourceLink name="jvx/server"
                  global="globalserver"
                  type="com.sibvisions.rad.server.Server" />
</DefaultContext>
```

The ResourceLink in the DefaultContext can also be defined in the META-INF/context.xml of the respective web application. This always depends on the configuration of the server or the web application. One possible example:

[context.xml](#)

```
<?xml version="1.0" encoding="UTF-8"?>
<Context>
  <ResourceLink name="jvx/server"
                global="globalserver"
                type="com.sibvisions.rad.server.Server" />
</Context>
```

To only make the server available for single web applications via JNDI, it should be configured as follows:

#### **META-INF/context.xml:**

##### context.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<Context>
  <Resource name="jvx/server" auth="Container"
            type="com.sibvisions.rad.server.Server"
            factory="org.apache.naming.factory.BeanFactory"/>
</Context>
```

We always recommend the configuration of the deployment descriptor, whether the server is provided for all applications or only for single web applications:

##### web.xml

```
<web-app ...>
  ...
  ...
  <resource-ref>
    <description>Object factory for Server instances.</description>
    <res-ref-name>jvx/server</res-ref-name>
    <res-ref-type>com.sibvisions.rad.server.Server</res-ref-type>
    <res-auth>Container</res-auth>
  </resource-ref>
</web-app>
```

In the case of a global server, the deployment descriptor would not have to be adapted, although it is recommended to better manage the utilized resources.

## **Note**

If the server is configured globally, we have to provide all of the application data – such as the rad

directory, .class files, etc. – also globally since the applications class loader is not used!

From:

<http://doc.sibvisions.com/> - **Documentation**

Permanent link:

[http://doc.sibvisions.com/jvx/server/security/server\\_jndi](http://doc.sibvisions.com/jvx/server/security/server_jndi)



Last update: **2020/06/25 10:47**