Table of Contents

Der Server für JVx Clients wird entweder in der aktuellen JVM instanziert oder am Applikationsserver wie z.B. Tomcat, JBoss. In der aktuellen VM genügt der Aufruf

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

um eine Server Instanz zu erstellen. Doch dadurch besteht die Gefahr das mehrere Server Instanzen erstellt werden (kann aber durchaus erwünscht sein). Um den Server als Singleton zu behandeln steht die Methode

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

zur Verfügung. Durch diese Verwendung wird zusätzlich versucht eine Server Instanz via JNDI anzusprechen.

Am Applikationsserver wird durch den ServletServer eine neue Instanz des Servers erstellt. Das wird durch den Aufruf von

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

erledigt. Wenn der Server nun als JNDI Resource zur Verfügung steht, übernimmt der Applikationsserver die Instanzierung des Servers. Dadurch könnte z.B eine Server Instanz für alle Applikationen eines Applikationsservers verwendet werden.

Die Konfiguration einer globalen JNDI Resource wird für den Applikationsserver Tomcat wie folgt durchgeführt:

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>
```

Der ResourceLink im DefaultContext kann auch im META-INF/context.xml der jeweiligen Web Applikation definiert werden. Das ist immer abhängig von der Konfiguration des Servers bzw. der Web Applikation. Ein mögliches Beispiel:

context.xml

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

Um den Server nur für einzelne Web Applikationen via JNDI bereitzustellen ist die Konfiguration wie folgt durchzuführen:

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>
```

Egal ob der Server für alle oder für einzelne Web Applikationen bereitgestellt wird. Es ist zu empfehlen den Deployment Deskriptor in allen Fällen zu konfigurieren:

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>
```

Im Falle eines globalen Servers müsste der Deployment Deskriptor nicht angepasst werden, ist jedoch zu empfehlen um die verwendeten Resourcen zu überblicken.

Hinweis

Wenn der Server global konfiguriert wird müssen auch alle Applikations-Daten wie z.B. rad Verzeichnis, .class Dateien, global zur Verfügung stehen, da nicht der Applikations ClassLoader verwendet wird!