

Table of Contents

Wie bereits bei den [Connections](#) erwähnt, werden Objekte bei der Übertragung zwischen Client- und Enterprise Tier serialisiert. Dazu wird jedoch nicht die Standard Java Serialisierung verwendet sondern spezielle, Technologie unabhängige, Serialisierungs Mechanismen.

Die sogenannten Serialisierer müssen das `ISerializer` Interface erfüllen.

Warum spezielle Serialisierer?

Die Kommunikation zwischen Java und C#, Flex usw. wäre mit Standard Java Serialisierung kaum möglich bzw. müsste die Java Serialisierung übernommen werden. Der Aufwand wäre enorm.

Vorhandene Serialisierungs Frameworks wie z.B. [Hessian](#), unterstützen nicht alle notwendigen Objekte wie z.B. `BigDecimal`.

Aus diesem Grund wurde ein eigener Serialisierer implementiert, der `UniversalSerializer`.

Durch die Definition von `ISerializer` können aber auch Frameworks wie `Hessian` in `JVx` integriert werden.

Anwendungsbeispiel

Wir implementieren einen Serialisierer der die Standard Java Serialisierung verwendet.

[JavaSerializer.java](#)

```
public class JavaSerializer implements ISerializer
{
    public Object read(DataInputStream in) throws Exception
    {
        ObjectInputStream ois = new ObjectInputStream(in);

        return ois.readObject();
    }

    public void write(DataOutputStream out, Object object) throws
Exception
    {
        ObjectOutputStream oos = new ObjectOutputStream(out);

        oos.writeObject(object);
    }
}
```

From:

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

Permanent link:

<https://doc.sibvisions.com/de/jvx/communication/serialization>



Last update: **2018/02/01 10:41**