FAQ Regelcode

Formatierungen in Java

Typische Formatierungsanforderungen, die man in Regeln immer wieder hat und jedes Mal mühsam nachschlagen muss.

Wir empfehlen, eine eigene Klasse (z.B. *NucletUtils*) zu erstellen, die immer wieder auftretende Anforderungen (wie unten beschriebene Formatierungen) als statische Funktionen (z.B. *public static String format(Date date)*) enthält. Eine solche schon vorbereitete Klasse kann alternativ auch mit dem Nuclet *Utils* geladen werden.

Datum formatieren

```
import java.text.SimpleDateFormat;
SimpleDateFormat df = new SimpleDateFormat("dd.MM.yyyy");
String s = df.format(new Date());
```

liefert heutiges Datum als formatierten String.

Zahl formatieren

```
import java.text.DecimalFormat;
import java.text.DecimalFormatSymbols;
DecimalFormat nf = new DecimalFormat("#,##0.0000");
nf.setDecimalFormatSymbols(DecimalFormatSymbols.getInstance(Locale.GERMAN));
String s = nf.format(new Double(1));
```

Führende Nullen ergänzen

```
int i = 9;
sNumber = String.format("%04d", i);
```

liefert "0009".

Dialoge (Input Specification)

```
JA/Nein - Dialog

if (showDialogEmail("Soll an die  Kundenemail:  " +email1 +" eine Email verschickt werden?")) {
// Email wird verschickt
}

private boolean showDialogEmail(String msg) throws BusinessException {
        if (InputContext.isSupported()) {
            Integer iValue =(Integer)InputContext.get("wEmailverschicken");
            if (iValue == null) {
                InputSpecification spec = new InputSpecification(InputSpecification.CONFIRM_YES_NO,
"wEmailverschicken", msg);
                throw new InputRequiredException(spec);
            } else {
                return (iValue.equals(1));
            }
        }
        else {
            throw new BusinessException("Nicht unterstützt.");
        }
    }
```