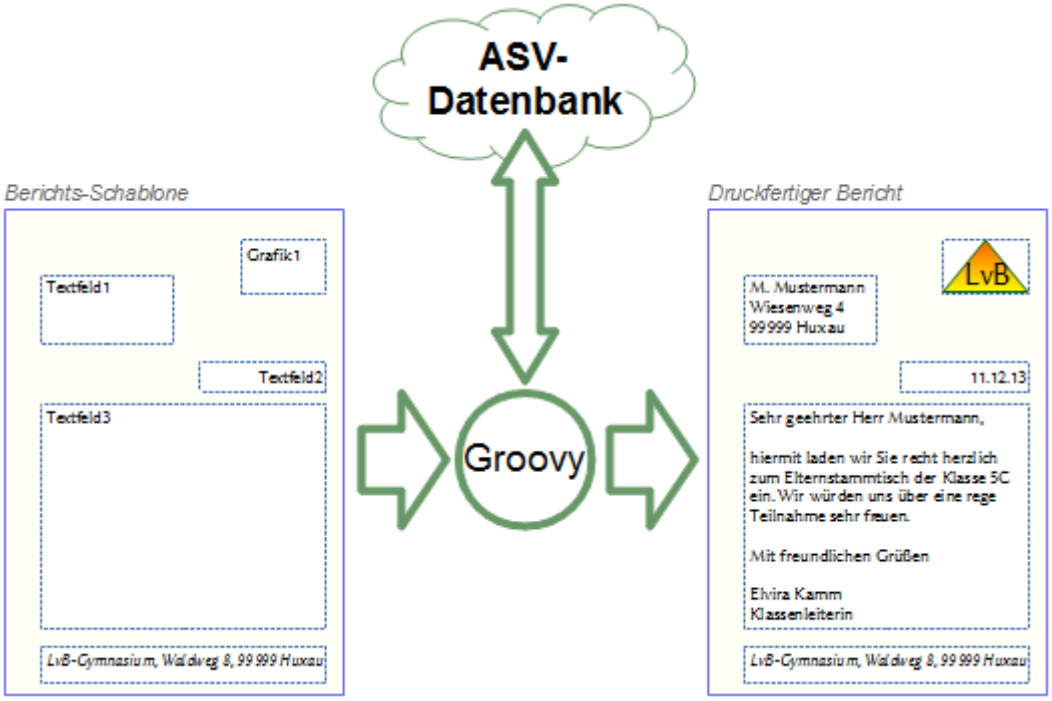
Groovy in ASV

Für einen ersten Umgang mit Groovy-Skripten ist es empfehlenswert, diese Seite durchzuarbeiten. Wer sich bereits mit den Grundlagen vertraut gemacht hat, findet in folgenden Kapiteln **weiterführende Informationen**:
Datum und Zeit

Grundsätzliches

Groovy ist eine universelle Skriptsprache für die JAVA-Virtual-Machine[1)]. Diese wird in ASV dazu verwendet, um die Inhalte von Textfeldern und Grafikrahmen beim Anwenden von Berichten zusammenzustellen oder zu verändern.



Event-Steuerung

Die Groovy-Skripte sind in den Eigenschaften **ONLOAD-EVENT**, **GET_FOCUS_EVENT** und **LOST_FOCUS_EVENT** von Textfeldern und Grafikrahmen gespeichert.
Der Ausführungszeitpunkt für jedes Skript hängt vom zugeordneten Event [2)] ab:

ONLOAD: beim Anwenden des Berichts (Zusammenstellung des evtl. nachträglich noch editierbaren Berichts)

Textfelder können so eingestellt werden, dass ihr Inhalt zwischen Zusammenstellung und Druck manuell editiert werden kann. In dieser Phase wird der vorläufig zusammengestellte Bericht auf dem Bildschirm angezeigt. In dieser Phase gibt es zwei auslösende Ereignisse:

GET_FOCUS: die Einfügemarke wird durch Mausklick oder Drücken der TAB-Taste in das Textfeld

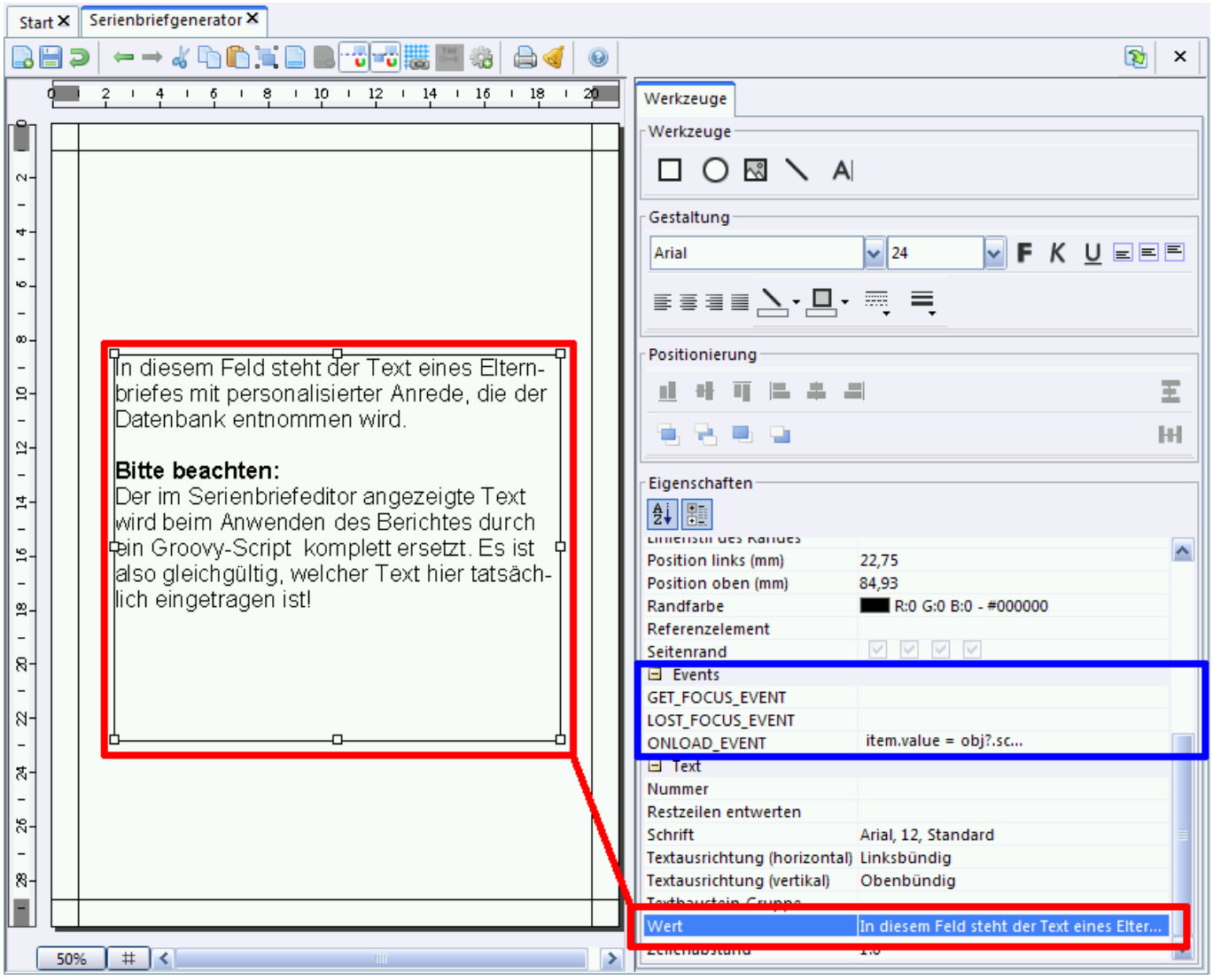
gesetzt
LOST_FOCUS: die Einfügemarke wird in ein anderes Textfeld versetzt, verlässt also dieses Textfeld

Grafiken und Texte

Die **Einbindung von Grafiken** ist relativ unkompliziert und bei Kenntnis des Datenbankfeldes mit einem einzelnen Befehl erledigt (nachzulesen bei der Beschreibung des Grafikwerkzeuges).
Dagegen ist die **Verarbeitung von Texten** in beinahe beliebig komplexem Umfang möglich und erfordert einige Grundkenntnisse, die im Folgenden erläutert werden.

Grundlagen bei Textfeldern

Der in einem Textfeld enthaltene Text ist in der Eigenschaft *Wert* gespeichert. Während des Editierens im Etikettengenerator oder Serienbriefeditor ist die Eingabe und Änderung sowohl in der Seitenansicht als auch in der Eigenschaft *Wert* veränderbar:



Bei der Verarbeitung des Textfeldinhaltes mit Groovy wird dieser Text jedoch in der Regel komplett ersetzt. Der zugehörige Skriptbefehl lautet beispielsweise:

```
item.value = "Neuer Textfeldinhalt";
```

`item` ist hierbei das Textfeld selbst, `item.value` beschreibt die Eigenschaft *Wert* des Textfeldes und damit seinen Inhalt.
Über die Zuweisung `item.value = …` wird der rechts notierte Inhalt in das Textfeld übertragen und der ursprüngliche Inhalt dementsprechend gelöscht.

Die Zuweisung kann natürlich neben festen Texten auch Berechnungen beinhalten. Hier ein Beispiel zum Eintrag des aktuellen Datums:

```
item.value = new Date().format('dd.MM.yyyy');
```

Es ist aber auch möglich, den ursprünglich im Textfeld eingetragenen Inhalt in der Berechnung rechts zu verwenden:

```
item.value = "vorausgestellter Text" + item.value + "angehängter Text";
```

Das aktuelle Objekt, für das ein Serienbrief oder ein bestimmtes Etikett erzeugt wird, ist über die vordefinierte Variable `obj` ansprechbar. Damit können Datenbankinhalte, die das Objekt betreffen, ausgelesen und verarbeitet werden. Die Schreibweise dazu ist `obj?.eigenschaft` bzw. `obj?.verweis?.eigenschaft`. Beispiel für einen Bericht aus dem Datenbereich *Schüler* (Klasse des Schülers im aktuellen Schuljahr).

```
item.value = obj?.schuelerStamm?.klasse;
```

Um den Inhalt eines *anderen* Textfeldes zu ändern, kann man folgenden Code verwenden:

```
doc.getItem("Textfeld1").setValue("Dies ist ein Test");
```

GRUNDREGELN

Groovy ist sehr intuitiv, es erkennt selbst, ob es Zahlen oder Text in der Variablenübergabe erhält, wobei Kommazahlen einen Dezimalpunkt erhalten (30.4). `item` ist hierbei das Textfeld selbst, `item.value` beschreibt die Eigenschaft *Wert* des Textfeldes und damit seinen Inhalt.
Über die Zuweisung `item.value = …` wird der rechts notierte Inhalt in das Textfeld übertragen und der ursprüngliche Inhalt dementsprechend gelöscht.

Das Komma ist für **Aufzählungen** (z.B. in Listen) vorgesehen.

```
liste1=[23.5, 22.5, 34.0, 22,7];
```

Jede **Anweisung** steht in einer Zeile oder wird am Ende mit einem Semikolon beendet.

```
item.value="leer";
```

Der ungeübte Anwender sollte nach jeder Anweisung ein Semikolon setzen und Klammern grundsätzlich paarweise schreiben und dann danach füllen. **Bedingungen** werden in runden Klammern erfasst. **Funktionen** erfordern Variablenübergaben in runden Klammern. **MehrfachAnweisungen** werden in kompakter Form mit geschweiften Klammern erfasst und verzichten damit auf das in älteren Programmiersprachen übliche begin und end.

```
 {a=5; item.value = 3.0*a; item.value=item.value+5}
```

Variablen- und Wertzuweisungen auch als Berechnung werden mit dem einfachen = belegt.

```
(a=3; b= "Berta"; c = a+b;)
```

Mit **+** und **-** kann man auch Zeichenketten verändern.

```
A= "ABC-Gymnasium";

B= "BC";

C=A-B;
```

C hat jetzt den Wert „A-Gymnasium“, die Zeichenkette „BC“ wurde gefunden und entfernt.

```
A= "ABC-Gymnasium";
B = "BlumenWeg 1";
C=A+ "\n" +B;
```

Die Ausgabe von C erfolgt durch den angeordneten Zeilenumbruch mit „\n“ zweizeilig.

Operatoren zum Vergleich sind unter Anderem das **Gleich ==**, das **Ungleich !=**, das **Kleiner <** und das **Größer >**.

Listen können ein- und mehrdimensional sein. Man definiert sie mit eckigen Klammern und ruft sie auch so auf. Eine Liste erhält Wertübergaben:

```
Liste1 = [20.2, 30.4, 45.8];
```

Aufruf einer Liste:

```
a=Liste1[1];
```

führt zur Belegung von a mit 30.4 (Liste1[0]=20.2 Liste1[2]= 45.8). Länge einer Liste: Die Länge einer Liste erfährt man mit

```
 Länge= Liste.size();
```

EINGABE VON DATEN

Die Datenbank von ASV ist sehr groß und entsprechend schwierig ist das Finden entsprechender Einträge. Die in ASV enthaltenen virtuellen Felder sind dafür aber sehr gute Wegweiser. Sie zeigen uns über aussagekräftige Namen den Weg zum Ziel. So findet man über die Verzweigung Lehrkraft - Person - Geburtsdatum schnell zum Eintrag vom Geburtsdatum.

Es gibt in der Datenbank nur drei große Arten von Daten. Zahlen als Einzeleinträge, Datumsangaben und Texte. Listen können Zahlen enthalten, die einen Dezimalpunkt haben. Listen können Zeichenketten enthalten.

Groovy erkennt das Format selbst. Das aktuelle Objekt (Lehrkraft, Schüler, Klasse..), für das ein Serienbrief oder ein bestimmtes Etikett erzeugt wird, ist über die vordefinierte Variable `obj` ansprechbar. Damit können Datenbankinhalte, die das Objekt betreffen, ausgelesen und verarbeitet werden. Die Schreibweise dazu ist `obj?.eigenschaft` bzw. `obj?.verweis?.eigenschaft`.

Leichter geht es für den Anfänger so: Ein Textfeld erhält ein **virtuelles Feld** mit Rechtsklick zugewiesen und über dieses den Echteintrag aus der Datenbank (Datenfeld einfügen, Lehrkräfte wählen, dann Pflichtunterricht dann Schuelernamen –> „Lehrkräfte.Unterricht.PflUnterricht.Schülernamen“ erzeugt.

Der Code enthält dann aber bereits das echte Datenziel:

```
item.value = obj?.lehrerStamm?.PflUnterrichtSchueler;
```

Den rechts vom IstGleichZeichen befindlichen Term weist man nun einer beliebigen eigenen Variable zu.

```
 Variable1 = obj?.lehrerStamm?.PflUnterrichtSchueler;
```

Bei Listen benötigt man noch eine kleine Änderung, damit Groovy weiß, dass dies eine Liste ist:

```
Liste2 = obj?.lehrerStamm?.getPflUnterrichtSchueler();
```

Wir verwenden zunächst eine eigene Variable, welche wir noch modifizieren wollen und setzten (nur für Listen) **get** und **()** an die entsprechenden Stellen. Danach können wir die Einzeleinträge in der Liste2 einzeln aufrufen und auch weiterverwenden.

Listen

```
Liste2 = obj?.lehrerStamm?.getPflUnterrichtSchueler();
```

Einzeldaten

```
klasse1 = obj?.schuelerStamm?.klasse;
```

ABFRAGEN

Skriptbefehl für Abfragen beruhen auf der Basis **IF ELSE** und werden in der **Bedingung rund ()** und in der **Anweisung geschweift {}** geklammert.

Ein Beispiel lautet dann:

```
if (a==3){b=5; c=b+1; item.value=c;} else {item.value="kein Wert vorhanden";}
```

Hierbei steht **if** für „ist etwas so, wie in der runden Klammer beschrieben“ dann führe aus, was in der geschweiften Klammer steht. **else** steht für andernfalls mache das, was in der zweiten geschweiften Klammer steht.

SCHLEIFEN

Der Code für Schleifen ist sehr vielfältig. Im Folgenden sei ein Beispiel vorgestellt, Groovy versteht aber alle Java-dialekte.

```
 For (Bedingung prüfen, Zähler setzen) {Anweisung};
```

Zählvariable ist im folgenden j, j wird zu Beginn mit 0 belegt und dann mit j++ in jeder Runde um 1 erhöht. Bedingung ist, dass j < als die Anzahl der Elemente in der Liste „zahlliste“ ist.

```
 for (j=0;j<zahlliste.size();j++){item.value=item.value+j;}
```

BEISPIELE

Datum :

```
item.value = new Date().format('dd.MM.yyyy');
```

Der folgende Quelltext gibt in einem Textfeld alle Oberstufenkurse mit Schülern aus. Sie können Ihn komplett in ein onload.event kopieren und testen.

```
schuelerliste = obj?.lehrerStamm?.getBesUnterrichtSchueler();
zahlliste = obj?.lehrerStamm?.getBesUnterrichtSchuelerzahl();
kursliste = obj?.lehrerStamm?.getBesUnterrichtFach();
artliste = obj?.lehrerStamm?.getBesUnterrichtUart();
Liste[0] = "Kein Einsatz";
a=1;
x=1;
```

```
item.value ="";
//Zahliste.size=anzahl der abgelegten Werte
for (j=0;j<zahlliste.size();j++) {
  //ist die Art mit "o" belegt
  if (artliste[j]=="o") {
    x++;
    //Welcher Kurs mit "o"
    liste[j] = kursliste[j]+ " " + artliste[j] + " " + zahlliste[j] + "\n";
    for (i=0;i<zahlliste[j];i++) {
      zahl=a+i;
      if (schuelerliste[zahl] != null) {
        //Welche Schüler?
        liste[j] = liste[j] + (i+1) + ". " + schuelerliste[zahl]+"\n";
}
}
    a=zahl+1;
}   else {liste[j]=" -- ";}
}
```

AUSGABE

Im ersten Textfeld setzt man mit folgendem Code fort:

```
if (x==1) {item.value="Kein Einsatz in der Oberstufe"} else item.value =
Liste[0]
```

Ausgabe in Folgefeldern Die anderen Listenelemente werden in die weiteren Textfelder geschrieben

Sehr empfehlenswert ist das Verwenden der Groovy WebConsole. Sie können damit noch weit besser testen(Großes Fenster - Fehlerausgaben prägnant mit Zeile und Nummer) und erhalten schlüssige Fehlermeldungen sofort auf den Bildschirm. Außerdem können Sir Ihre Quelltexte an jedem Rechner modifizieren und müssen dazu nicht zwingend in ASV arbeiten.

Legen Sie sich dazu vorher eigens belegte Wertzuweisungen in der folgenden Form als **Ersatzeingabe** zu:

```
 schuelerliste=["Otto Einstein", "Bea Bärlauch", "Susi Sonnschein",....,
"Erwin Windfang", "Fritz Schleicher"];
```

```
 kursliste = ["Deutsch", "Englisch", "Mathematik"];
```

```
 zahlliste = [5, 6, 2, 4];
```

```
 artliste = ["o", "p", "o", "x"];
```

```
 Liste[0] = "Kein Einsatz";
```

Schreiben Sie nun Ihre **Verarbeitung** in der Konsole und legen Sie die Einzelausgaben, die sie wünschen in einer neuen Liste ab. Da sich ASV die im ersten Textfeld belegten Variablen auch in den anderen Textfeldern merkt, ist die Organisation der **AUSGABE** leicht zu bewerkstelligen. In das erste

Textfeld kopieren Sie die komplette Verarbeitung aus der Groovy-Konsole und lassen die **Eingabe**, so wie oben beschrieben.

```
zahlliste = obj?.lehrerStamm?.getBesUnterrichtSchuelerzahl();
kursliste = obj?.lehrerStamm?.getBesUnterrichtFach();
artliste = obj?.lehrerStamm?.getBesUnterrichtUart();
```

.. **Hier steht der Quelltext für die Verarbeitung gefolgt von Ausgabe1 in Textfeld 1** ..

ersetzt wird dabei in der Ausgabe das **println** mit **item.value**

```
 item.value =item.value + Liste[0];
```

Übertragen Sie ihre folgenden Listenelemente je nach Index an die durchnummerierten Textfelder des Serienbriefes mit

Das ist die AUSGABE in den folgenden Textfeldern

```
item.Value = item.value + Liste[index];
```

Einfacher Komplettquelltext zum Ausprobieren in der Web console:

```
lehrerliste=["Otto Einstein", "Bea Bärlauch", "Susi Sonnschein", "Erwin
Windfang"];
kursliste = ["Deutsch", "Englisch", "Mathematik", "Deutsch"];
zahlliste = [5, 6, 2, 4];
artliste = ["o", "p", "o", "x"];
liste=[];
for (j=0;j<zahlliste.size();j++) {
      liste[j] = kursliste[j]+ " " + artliste[j] + " " + zahlliste[j] +
"\n";
    }

for (j=0;j<zahlliste.size();j++) { println(liste[j]);}
```

Ergebnis der Konsole in der Anzeige:

Deutsch o 5

Englisch p 6

Mathematik o 2

Deutsch x 4

Die letzte Schleife listet alle Elemente aus dem Array Liste auf. Diese kann man über Index in die einzelnen Textfelder kopieren mit

Textfeld1:

```
item.value =item.value + Liste[0];
```

Textfeld2:

```
item.value =item.value + Liste[1];
```

usw.

Die Ausgabe in mehreren Textfeldern erlaubt dann, das Dokument auch unterschiedlich zu formatieren.

—

Den Quelltext zum Datum und dessen Formatierung finden Sie hier:

http://www.asv.bayern.de/wiki/berichte/werkzeuge/groovy/datum_und_zeit

1)
JAVA bietet eine plattformübergreifende virtuelle Umgebung, so dass ASV in unterschiedlichen Betriebssystemen zur Verfügung steht.
2)
Ein *Event* ist ein Ereignis, das die Ausführung eines Skriptes anstößt.