Einführung in Virthos Master

Stand 11.09.2008

Virthos Systems GmbH
www.virthos.net

Inhalt

Über dieses Handbuch

Dieses Handbuch beschreibt die Besonderheiten von Virthos Master, einer erweiterten Version des Content-Management-Systems Virthos Pro. Es wendet sich an Template-Designer und Anwendungsentwickler, die mit Hilfe von Virthos dynamische Internetauftritte oder Webapplikationen entwickeln wollen. Um die Beschreibungen zu verstehen und nachvollziehen zu können, sollten Sie bereits Erfahrungen mit dem Entwickeln von Templates für Virthos Pro gesammelt haben und möglichst auch über allgemeine Programmierkenntnisse, zum Beispiel in PHP, verfügen.

Die speziellen Funktionen von Virthos Master werden in diesem Handbuch nur allgemein und beispielhaft beschrieben. Einzelheiten zu den einzelnen Platzhaltern und Anweisungen finden Sie im Virthos-Referenzhandbuch.

Virthos Master im Überblick

Die "Master"-Version von Virthos ist, kurz gesagt, eine erweiterte "Pro"-Version. Ein Webauftritt, der mit der "Pro"-Version eingerichtet wurde, lässt sich problemlos mit der "Master"-Version weiterbetreiben, ohne dass die Besucher oder die Redakteure einen Unterschied bemerken. Allerdings bietet die "Master"-Version Ihnen als Template-Entwickler zusätzliche Möglichkeiten, um den Webauftritt mit neuen Funktionen auszustatten. Insbesondere lassen sich bestimmte Abläufe, wie das Erstellen von Seiten oder das Anlegen neuer Benutzer, automatisieren, und Sie können eigenen PHP-Code uneingeschränkt in Virthos-Templates einbinden.

Ein Wort zur Warnung: Einige der zusätzlichen Möglichkeiten, die Virthos Master bereitstellt, können, wenn sie in eigenen Templates benutzt werden, die Sicherheit eines Webauftritts beeinträchtigen. Sie sollten sich daher mit den grundlegenden Sicherheitsregeln vertraut machen, die für alle Arten der serverseitigen Programmierung gelten. Beachten Sie, dass die möglichen Sicherheitslücken nur entstehen können, wenn Sie bestimmte Anweisungen in Ihren Templates verwenden; solange Sie die Templates unverändert lassen, wird durch den Umstieg von Virthos Pro auf Virthos Master die Sicherheit nicht beeinträchtigt.

Inline-Aktionen mit vtDo

Virthos Master stellt eine VirthosTalk-Anweisung zur Verfügung, mit der sich Vorgänge, die normalerweise durch das Absenden eines Formulars angestoßen werden, auch "inline" – also einfach durch das Abrufen einer Seite – ausführen lassen. Diese Anweisung sieht folgendermaßen aus:

```
<!--{{vtDo: -act="xxx", yyy}}-->
```

Das "xxx" müssen Sie durch den Namen der Aktion ersetzen, die ausgeführt werden soll, das "yyy" durch zusätzliche Angaben, die für das Ausführen der Aktion erforderlich sind (und die in vielen Fällen auch durch das Absenden eines Formulars übergeben werden könnten).

Ein einfaches Beispiel könnte so aussehen:

```
<!--{{vtDo: -act="create", -target="//", -template="logentry"}}-->
```

Wenn Sie diese Anweisung in ein Template einbinden und in Virthos eine Seite auf Basis dieses Templates anlegen, wird bei jedem Aufruf dieser Seite ein "Logeintrag" erzeugt. Logeintrag bedeutet in diesem Fall einfach, dass auf der obersten Virthos-Ebene eine neue Seite vom Typ "logentry" erzeugt wird. Auf diese Weise sammelt sich schnell eine Menge "Abfall" an, den Sie mit Hilfe der folgenden Anweisungen aber schnell wieder beseitigen können:

```
<!--{{vtUse://}}-->
    <!--{{vtLoop: logentry}}-->
        <!--{{vtDo: -act="trash", -obj="{vtID}"}}-->
    <!--{{vtEndLoop}}-->
<!--{{vtEndUse}}-->
```

Die vtUse-Anweisung macht die oberste Virthos-Ebene zum aktuellen Kontext. Mit vtLoop werden dann alle Seiten vom Typ "logentry" durchlaufen, die sich auf dieser Ebene befinden. Und die vtDo-Anweisung verschiebt diese Seiten dann in den Papierkorb.

Besonders hilfreich ist vtDo, wenn man Sessionvariablen speichern möchte, ohne dass ein Formular abgesendet oder ein Hyperlink angeklickt wird:

```
<!--{{vtDo: -act="updateSession", Variable1="Wert1", Variable2="Wert2" }}-->
```

Grundsätzlich können Sie mit vtDo alle Aktionen verwenden, die auch als Formularaktionen zur Verfügung stehen. Welche dies sind und welche Parameter Sie jeweils angeben müssen, ist im dritten Teil des Virthos-Benutzerhandbuches im Kapitel "Formularverarbeitung"

beschrieben. Zusätzlich zu den dort aufgeführten Aktionen stellt Virthos Master noch weitere Aktionen zur Verfügung:

Aktion	Beschreibung	Parameter
setCookie	speichert Variablen in einem Cookie	siehe den Abschnitt über Cookies weiter hinten
importCSV	importiert eine CSV-Datei, erstellt für jede Zeile eine Seite in Virthos	`-file` (CSV-Datei, die per Formular hochgeladen wurde; diese muss in der ersten Zeile die Namen der Platzhalter enthalten, die der jeweiligen Spalte entsprechen), `-template` (Name des Seitentyps für die erstellten Seiten)
addUserToGroup	fügt einen Benutzer zu einer Gruppe hinzu	`-user` (Name des Benutzers), `-group` (Name der Gruppe)

Beachten Sie: Anders als bei Formularaktionen prüft Virthos bei einer Inline-Aktion nicht, ob der aktuelle Benutzer die nötigen Rechte besitzt, um die Aktion auszuführen. Diese Prüfung müssen Sie daher, wenn sie gewünscht ist, selbst in Form von vtIf-Bedingungen in das jeweilige Template einbauen.

Ereignisgesteuerte Methoden

Mit Hilfe der vtDo-Anweisung lassen sich Vorgänge automatisch ausführen, wenn eine bestimmte Seite aufgerufen wird. Das ist in vielen Fällen hilfreich und ausreichend, aber manchmal würde es die Sache erleichtern, wenn man die Ausführung nicht an das Aufrufen einer Seite, sondern an andere Ereignisse koppeln könnte. Virthos Master stellt dafür einen Mechanismus bereit, den wir als "ereignisgesteuerte Methoden" bezeichnen. Was es damit auf sich hat, wird am besten an einem Beispiel deutlich.

Angenommen, Sie haben ein Gästebuch eingerichtet, das aus den Templates *gbook.html* und *gbookentry.html* besteht. Nun möchten Sie, dass bei jedem neuen Gästebucheintrag eine Benachrichtigung per E-Mail versendet wird. Um dies zu erreichen, müssten Sie eine Datei erstellen, die folgende Anweisung enthält:

```
<!--{{vtDo: -act="mail", -to="info@domain.de",
        -subject="Neuer Gästebuch-Eintrag", -message="{Eintrag}"}}-->
```

Damit diese Anweisung automatisch ausgeführt wird, wenn ein neuer Eintrag hinzugefügt wurde, müssen Sie die Datei einfach nur richtig benennen. Der korrekte Name wäre in diesem Fall: `gbookentry._oncreate.html`. Achten Sie auf den Unterstrich hinter dem Punkt!

Wann immer eine neue Seite angelegt wird, prüft Virthos Master, ob es für den betreffenden Seitentyp eine "oncreate"-Methode gibt. Wenn dies der Fall ist, werden die Anweisungen in dieser Datei sofort nach dem Erstellen der Seite ausgeführt. In einer solchen Methode können Sie alle Platzhalter und Anweisungen verwenden, die Virthos zur Verfügung stellt. Die obige Methode ließe sich zum Beispiel folgendermaßen erweitern:

```
<!--{{vtIf: {Eintrag} .cn. viagra }}-->
    <!--{{vtDo: -act="trash", -obj="{vtID}"}}-->
<!--{{vtElse}}-->
    <!--{{vtDo: -act="mail", -to="info@domain.de",
            -subject="Neuer Gästebuch-Eintrag", -message="{Eintrag}"}}-->
<!--{{vtEndIf}}-->
```

Dies ist ein sehr einfacher Spam-Filter, der die grundsätzlichen Möglichkeiten verdeutlicht: Wenn in dem Gästebucheintrag das Wort "viagra" vorkommt, wird die gerade erstellte Seite sofort in den Papierkorb verschoben. Nur wenn das Wort "viagra" nicht vorkommt, erfolgt eine E-Mail-Benachrichtigung.

Die Anweisungen in einer oncreate-Methode werden im Kontext der gerade erstellten Seite ausgeführt, darum können Sie in dem obigen Beispiel direkt den Platzhalter `{Eintrag}`

verwenden (wobei wir hier stillschweigend davon ausgegangen sind, dass in dem Formular zum Erstellen des Gästebucheintrags das betreffende Formularfeld ebenfalls "Eintrag" heißt).

Virthos Master kennt neben dem Erstellen von Seiten noch andere Ereignisse, für die man entsprechende Methoden bereitstellen kann. Hier sind alle Ereignisse im Überblick:

Methode	wird ausgeführt,...
oncreate	wenn eine Seite des betreffenden Typs erstellt wurde
onupdate	wenn eine Seite des betreffenden Typs verändert wurde
onchange	wenn eine Seite des betreffenden Typs erstellt oder verändert wurde
onadd	wenn zu einer Seite des betreffenden Typs eine neue Subseite hinzugefügt wurde (entweder durch Erstellen einer neuen Seite oder durch Verschieben einer bestehenden Seite mittels Ausschneiden/Einfügen)

Die Ereignisse überschneiden sich teilweise, so dass ein einzelnes Ereignis mehrere Methoden auslösen kann. Beim Erstellen einer neuen Seite wird beispielsweise für die neue Seite die "oncreate"- und die "onchange"-Methode und für die Containerseite die "onadd"-Methode ausgeführt. Die Reihenfolge der Ausführung entspricht der Folge, in der die Methoden in der Tabelle aufgeführt sind.

Wichtig zu beachten: Grundsätzlich werden ereignisgesteuerte Methoden auch dann ausgeführt, wenn das Ereignis durch eine Inline-Aktion ausgelöst wurde, wenn Sie also beispielsweise mit vtDo eine neue Seite erstellt haben. Um Endlosschleifen zu verhindern, gilt dies allerdings nicht für Inline-Aktionen, die sich selbst innerhalb einer ereignisgesteuerten Methode befinden.

Eigene Datenbankabfragen

Virthos Master stellt Anweisungen und Platzhalter zur Verfügung, um eigene mySQL-Datenbankabfragen in Templates einzubauen. Das folgende Beispiel zeigt, wie das grundsätzlich aussieht:

```
<!--{{vtDBSelect: SELECT field1, field2 AS testname, field3 FROM dbtable
                  WHERE somefield = '{vtGet:VirthosVariable}' }}-->
    <!--{{vtIf: {vtDBCountSelection}.gt.0 }}-->

        <!--{{vtDBLoop}}-->
            <p>
                Feld 1: {{vtDBField:field1}}<br>
                Feld 2: {{vtDBField:testname}}<br>
                Feld 3: {{vtDBField:field3}}
            </p>
        <!--{{vtEndDBLoop}}-->

    <!--{{vtEndIf}}-->
<!--{{vtEndDBSelect}}-->
```

In der vtDBSelect-Anweisung kann man alle Funktionen und Schreibweisen verwenden, die der mySQL-Server innerhalb von SELECT-Anweisungen unterstützt. Andere mySQL-Befehle (z.B. "UPDATE") sind nicht zulässig – um diese innerhalb eines Virthos-Templates zu verwenden, muss man auf Standard-PHP-Funktionen zurückgreifen (vgl. den Abschnitt "Einbindung von eigenem PHP-Code").

Eine Datenbankabfrage, die Sie innerhalb von vtDBSelect angeben, darf Virthos-Platzhalter enthalten, wie im obigen Beispiel gezeigt. Man kann auch die Abfrage als Ganzes durch einen Platzhalter angeben, wenn dieser die komplette Syntax enthält.

Eigenen PHP-Code einbinden

Mit vtCalc steht in allen Virthos-Versionen eine Möglichkeit zur Verfügung, um eigenen PHP-Code in Templates einzubinden. Diese Möglichkeit ist allerdings auf einfache PHP-Ausdrücke beschränkt, komplexere Anweisungen oder gar Anweisungsblöcke lassen sich mit vtCalc nicht verwirklichen. Virthos Master stellt daher eine zusätzliche VirthosTalk-Anweisung zur Verfügung, um PHP-Code in Templates einzubinden, und zwar beliebigen Code, ohne jegliche Einschränkungen. Ein einfaches Beispiel sieht so aus:

```
<!--{{vtCode: echo "Hello World!"; }}-->
```

Man notiert einfach hinter dem Doppelpunkt den gewünschten PHP-Code, so wie man ihn auch in einem PHP-Skript schreiben würde. Beachten muss man dabei nur, dass innerhalb des Codes nicht zwei schließende geschweifte Klammern unmittelbar aufeinanderfolgen, denn dies würde vom Virthos-Parser als das Ende der vtCode-Anweisung interpretiert. Es genügt, wenn ein Leerzeichen oder ein Zeilenumbruch zwischen den geschweiften Klammern steht.

Innerhalb von vtCode sind beliebige PHP-Anweisungsblöcke erlaubt, die sich auch über mehrere Zeilen erstrecken dürfen:

```
<!--{{vtCode:
    for( $i=1; $i<10; $i++ ) {
        echo "Hello World!";
    }
}}-->
```

Bevor man PHP-Code in ein Virthos-Template einbindet, empfiehlt es sich, ihn innerhalb eines herkömmlichen PHP-Skripts zu testen. Virthos gibt bei eingebundenem PHP-Code nämlich keinerlei Fehler- oder Warnmeldungen aus, was die Fehlersuche schwierig gestaltet. In jedem Fall ist es wichtig zu beachten, dass auch eine einzelne PHP-Anweisung in vtCode mit einem Semikolon abgeschlossen werden muss.

Da vtCode beliebige PHP-Anweisungen erlaubt, ist es nicht möglich (wie in vtCalc), dass man Virthos-Platzhalter einfach in den Code einsetzt. Das folgende Beispiel macht den Unterschied deutlich:

```
<!--{{vtCalc: substr('{Headline}', 0, 3); }}-->
{{vtResult}}
```

Diese Anweisung würde die ersten drei Zeichen ausgeben, die im Feld "Headline" gespeichert sind, wogegen

```
<!--{{vtCode: echo substr('{Headline}', 0, 3); }}-->
```

stets "{He" ausgeben würde, unabhängig vom Inhalt des Headline-Feldes.

Um innerhalb des PHP-Codes auf Virthos-Daten zugreifen zu können, steht eine API in Form einer PHP-Klasse zur Verfügung, die in dem genannten Beispiel so verwendet werden könnte:

```
<!--{{vtCode:
    $v = new virthos;
    echo substr( $v->vtpage('Headline'), 0, 3); }}-->
}}-->
```

Die Virthos-Klasse braucht innerhalb eines Templates nur einmal initialisiert zu werden. Neben der vtpage-Funktion, mit der sich seitenbezogene Inhalte ausgeben lassen, steht beispielsweise auch eine vtget-Funktion zur Verfügung, um auf Variablen zuzugreifen die zuvor mit der {{vtSet:...}}-Anweisung gesetzt wurden:

```
<!--{{vtSet: Name="{vtUser:name}" }}
<!--{{vtCode: $user = $v->vtget('Name'); }}-->
```

Umgekehrt lassen sich innerhalb von vtCode Variablen setzen, die außerhalb mit dem vtGet-Platzhalter ausgegeben werden können:

```
<!--{{vtCode: $v->vtset( array('Ergebnis'=>'okay') ); }}-->
{{vtGet:Ergebnis}}
```

Jede API-Funktion entspricht einem Platzhalter oder einer Anweisung des VirthosTalk-Sprachumfangs, sowohl in der Benennung als auch in der Struktur der Parameter. Wo in VirthosTalk nur ein einzelner Parameter vorgesehen ist, wird dieser als String übergeben, ansonsten erwartet die API-Funktion ein indiziertes oder assoziatives Array als Parameter. Eine Sonderstellung nimmt die bereits beschriebene Funktion `vtpage()` ein, für die es in VirthosTalk kein Gegenstück gibt.

Nicht alle VirthosTalk-Platzhalter und -Anweisungen stehen in der API zur Verfügung. Welche Funktionen verfügbar sind, können Sie der folgenden Tabelle entnehmen:

Funktion	Beschreibung
`vtpage( $str )`	Gibt den Inhalt des seitenbezogenen Platzhalters `$str` aus. Seitenbezogen sind alle freien Platzhalter sowie die Platzhalter "vtComment", "vtCreationDate", "vtCreationUser", "vtID", "vtModificationDate", "vtModificationUser", "vtName", "vtPackage", "vtPageType" und "vtPathIDs".

Funktion	Beschreibung
`vtget( $str )`	Gibt den Wert der Variablen `$str` aus, die mittels `vtSet`-Anweisung gesetzt wurde.
`vtsession( $str )`	Gibt den Wert der Sessionvariablen `$str` aus, die mittels `updateSession`-Aktion gesetzt wurde.
`vtglobal( $str )`	Gibt den Wert der globalen Variablen `$str` aus.
`vtdo( $action, $arr )`	Führt die Inline-Aktion `$action` aus. Der zweite Parameter ist ein assoziatives Array mit den Parametern, die für die Aktion erforderlich sind.
`vtset( $arr )`	Setzt eine oder mehrere Variablen. Die Namen der Variablen entsprechen den Schlüsselnamen des assoziativen Arrays `$arr`, die Variablenwerte den Arraywerten.
`vtlink( $arr )`	Gibt eine URL aus, die auf eine andere Virthos-Seite verweist. Im Array `$arr`, das indiziert oder assoziativ sein kann, müssen die Parameter übergeben werden, die beim vtLink-Platzhalter dokumentiert sind.

PHP-Bibliotheken einbinden

Wollen Sie innerhalb von vtCode (oder vtCalc) auf Klassen und Funktionen zugreifen, die in externen Bibliotheken definiert sind, können Sie diese Bibliotheken mit Hilfe der `vtLoad`-Anweisung ins Template einbinden:

```
<!--{{vtLoad: classes/meine_funktionen.php }}-->
```

Dies entspricht der Anwendung eines `include_once`-Befehls in PHP. Beachten Sie, dass eine relative Pfadangabe bei `vtLoad` immer vom Speicherort der virthos.php-Datei aus aufgelöst wird und nicht, wie zum Beispiel bei `vtInclude`, von der Template-Datei aus.

Seiteninhalte zwischenspeichern

Verwendet man in einem Template mehrere vtSelect- oder vtLoop-Anweisungen, kann dies die Geschwindigkeit, mit der die Seite aufbereitet wird, deutlich herabsetzen. Um dem entgegenzuwirken, stellt Virthos Master einen Mechanismus bereit, mit dem sich fertig aufbereitete Listen oder sonstige Seitenbestandteile auf der Festplatte zwischenspeichern ("cachen") lassen.

Die Vorgehensweise, um diesen Mechanismus einzusetzen, soll anhand eines einfachen Beispiels beschrieben werden. Nehmen wir an, Sie hätten ein Template namens *newslist.html*, das folgendermaßen aussieht:

```
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>Newsliste</title>
    ...
</head>
<body>

    <div id="header"> ... </div>

    <div id="content">
        <!--{{vtSelect:-origin="//News", -type="news",
                -sortfield="Datum",-sortorder="descending"}}-->
        <!--{{vtLoop:-size="30"}}-->
            <h2>{{Headline}}</h2>
            <p>{{Datum}} &mdash; {{Intro}} <a href="{{vtLink}}">mehr</a></p>
        <!--{{vtEndLoop}}-->
        <!--{{vtEndSelect}}-->
    </div>

    <div id="footer"> ... </div>

</body>
</html>
```

Nehmen wir ferner an, Sie haben in Virthos eine Seite auf Basis dieses Templates angelegt, und diese Seite hat die Nummer 123. Da die Website hohe Besucherzahlen aufweist, aber nur gelegentlich News hinzukommen, ist es sinnvoll, den vtSelect-Block in eine Cachedatei auszulagern, die lediglich einmal am Tag aktualisiert wird. Gehen Sie dazu folgendermaßen vor:

1. Erstellen Sie ein Template für den vtSelect-Block.

Kopieren Sie den gesamten Block von `<!--{{vtSelect:...` bis `{{vtEndSelect}}-->` in ein eigenes Template namens *newsblock.html*. Legen Sie dieses nicht direkt in das Verzeichnis, in dem newslist.html liegt, sondern in ein Unterverzeichnis namens *blocks*. Dadurch bleibt die Übersichtlichkeit gewahrt.

2. Legen Sie das Cache-Verzeichnis an.

Sie brauchen ein Verzeichnis, in dem die Cache-Dateien erzeugt werden. Legen Sie dazu (falls noch nicht vorhanden) im *data*-Verzeichnis ein Unterverzeichnis namens *cache* an. Die Zugriffsrechte müssen so gesetzt sein wie bei den übrigen Unterverzeichnissen, die sich im *data*-Verzeichnis befinden. Der Webserver muss in dieses Verzeichnis schreiben und daraus lesen können.

3. Erzeugen Sie die Cache-Datei.

Um die Cache-Datei zu erzeugen, müssen Sie die *virthos.php*-Datei in Ihrem Webbrowser mit speziellen URL-Parametern aufrufen:

```
http://.../virthos.php?-pg=123&-met=blocks/newsblock.html&-out=newsblock.html
```

Der -pg-Parameter enthält die Nummer der Seite, ín deren Kontext das Template interpretiert werden soll. Im Normalfall geben Sie hier die Seite an, in die der Cacheblock eingebunden wird. In unserem Beispiel könnten wir die Seitenangabe auch weglassen, weil das Template keine seitenabhängigen Platzhalter oder Anweisungen enthält; aber schaden kann diese Angabe nicht.

Der -met-Parameter enthält den Pfad zur Cache-Templatedatei, ausgehend von dem Verzeichnis, in dem die gewöhnlichen Templates liegen.

Der -out-Parameter ist das eigentlich Besondere an dem Aufruf. Dieser Parameter steht nur in Virthos Master zur Verfügung und weist Virthos an, die aufbereiteten Inhalte nicht an den Webbrowser zu liefern, sondern sie in eine Datei zu schreiben, in diesem Fall in die Datei *newsblock.html*.

Wenn Sie diese URL zum ersten Mal aufrufen, erzeugt Virthos im Cache-Verzeichnis automatisch ein Unterverzeichnis namens *blocks* und darin die Datei *newsblock.html*.

4. Binden Sie die Cache-Datei ins Seitentemplate ein.

Damit Sie die Vorteile der Cache-Datei auch wirklich nutzen, müssen Sie nun noch das ursprüngliche Template anpassen. Entfernen Sie dazu den vtSelect-Block und fügen Sie an seiner Stelle eine vtInclude-Anweisung ein:

```
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>Newsliste</title>
    ...
</head>
<body>

    <div id="header"> ... </div>

    <div id="content">
        <!--{{vtInclude:/virthos/data/blocks/newsblock.html}}
    </div>

    <div id="footer"> ... </div>

</body>
</html>
```

Die vtInclude-Anweisung muss den absoluten Pfad der Cache-Datei enthalten, also ausgehend vom Wurzelverzeichnis des Webauftritts. Das obige Beispiel geht davon aus, dass Virthos in einem Unterverzeichnis namens *virthos* installiert ist. Dies müssen Sie gegebenenfalls an Ihre Gegebenheiten anpassen.

5. Optimieren Sie das Cache-Template

Wenn Sie der Anleitung bis hierher gefolgt sind, haben Sie bereits ein funktionierendes Cache-System vorliegen. Sie können sich davon überzeugen, indem Sie die Seite mit der Newsliste im Webbrowser aufrufen. Es gibt allerdings noch ein kleines Problem, das sich zeigt, wenn Sie die Cache-Datei *cache/blocks/newsblock.html* in einem Texteditor öffnen. Sie werden nämlich feststellen, dass die Hyperlinks, die sich darin befinden, sessionbezogene Parameter enthalten (-ses und -cod). Wann immer jemand auf einen dieser Links klickt, versucht Virthos die Session wiederherzustellen, innerhalb derer die Cache-Datei erzeugt wurde. Mit anderen Worten: Der Besucher verlässt "seine" Session, und verliert damit alle sessionbezogenen Informationen, die in Virthos bis dahin gespeichert worden sind. Ein Redakteur zum Beispiel, der sich bei Virthos angemeldet hat, würde beim Klick auf einen solchen Link seinen Redakteursstatus verlieren und könnte im Virthos-Manager nicht mehr richtig weiterarbeiten.

Damit die Hyperlinks in jeder Session die richtigen Informationen weitertransportieren, müssen sie in jeder Session anders aussehen. Mit anderen Worten: Sie dürfen nicht statisch in den Cache-Block eingebunden werden, sondern müssen beim Aufruf der Seite dynamisch erzeugt werden. Da der Cache-Block wie eine gewöhnliche Include-Datei eingebunden wird, darf er auch VirthosTalk-Platzhalter enthalten, die Frage ist nur, wie wir verhindern, dass

diese Platzhalter beim Erzeugen der Cache-Datei bereits durch statischen Code ersetzt werden.

Virthos Master stellt zu diesem Zweck einen speziellen Platzhalter namens "vtOut" zur Verfügung. In unserem Beispiel könnten wir ihn folgendermaßen einsetzen:

```
<p>{{Datum}} &mdash; {{Intro}} <a href="{{vtOut:vtLink}}">mehr</a></p>
```

Wenn Virthos beim Erzeugen der Cache-Datei auf einen vtOut-Platzhalter stößt, bindet es alles, was hinter dem Doppelpunkt folgt, als Platzhalter in die Cache-Datei ein. Das Ergebnis könnte zum Beispiel so aussehen:

```
<p>12.04.2008 &mdash; Dies ist ein Introtext. <a href="{{vtLink}}">mehr</a></p>
```

Während also die Platzhalter `{{Datum}}` und `{{Uhrzeit}}` durch statische Inhalte ersetzt wurden, bleibt an der Stelle, an der `{{vtOut:...}}` stand, ein vtLink-Platzhalter stehen. Und damit werden die Hyperlinks tatsächlich erst erzeugt, wenn jemand die Newsliste aufruft, so dass dann alle Sessioninformationen erhalten bleiben.

Unser Problem ist damit aber noch nicht ganz gelöst, denn `{{vtLink}}` verweist, wenn nichts anderes angegeben ist, immer auf die Seite, die den aktuellen Kontext bildet. Innerhalb des vtLoop-Blocks, so wie er im Template ursprünglich vorlag, wurde der Kontext richtig gesetzt, aber dieser Block existiert ja in der Cache-Datei nicht mehr. Hier würde `{{vtLink}}` bei allen News in der Liste auf dieselbe Seite verweisen, nämlich die Newsliste selbst.

Die Lösung besteht darin, das Verweisziel im vtLink-Platzhalter explizit anzugeben:

```
<p>{{Datum}} &mdash; {{Intro}} <a href="{{vtOut:vtLink: {vtID} }}">mehr</a></p>
```

Das Ergebnis könnte dann in der Cache-Datei so aussehen:

```
<p>12.04.2008 &mdash; Dies ist ein Introtext. <a href="{{vtLink: 245 }}">mehr</a></p>
```

Damit haben wir unser Ziel erreicht: Die Cache-Datei enthält keine sessionbezogenen Informationen mehr, und die Hyperlinks werden bei jedem Seitenabruf korrekt gebildet.

6. Automatisieren Sie die Cache-Erzeugung

Wenn die Redakteure öfter neue Einträge für die Newsliste erstellen, ist das manuelle Erzeugen der Cache-Datei auf Dauer etwas lästig, und es besteht die Gefahr, dass dies vergessen wird. Daher empfiehlt es sich, diesen Vorgang zu automatisieren. Eine einfache Lösung besteht darin, einen Onlinedienst wie cronjob.de zu verwenden. Dort lassen sich URLs hinterlegen, die in bestimmten Zeitabständen automatisch aufgerufen werden. Wenn Sie dort die

URL aus Schritt 3 eintragen und beispielsweise jede Nacht um 3:00 Uhr aufrufen lassen, haben Sie mit wenig Aufwand eine automatische tägliche Cache-Aktualisierung verwirklicht.

Eine Alternative, die allerdings voraussetzt, dass Sie auf dem Webserver über erweiterte Zugriffsrechte verfügen, besteht darin, ein Shell-Skript für die Cache-Erzeugung einzurichten und dieses in die Cron-Tabelle des Systems einzutragen. Sie können dabei mit den Unix-Befehlen `wget` oder `curl` arbeiten und die URL genauso angeben, wie in Schritt 3 beschrieben. Effizienter ist es allerdings, das Virthos-PHP-Skript direkt über die Kommandozeile anzusprechen. Der Befehl dazu könnte beispielsweise so aussehen:

```
/usr/bin/php virthos.php -pg=123 -met=blocks/newsblock.html -out=newsblock.html
```

Falls sich der PHP-Interpreter nicht im Verzeichnis */usr/bin/* befindet, müssten Sie die Pfadangabe entsprechend anpassen. Und wenn sich das Shell-Skript, das diesen Befehl enthält, nicht im gleichen Verzeichnis befindet wie die *virthos.php*-Datei, müssten Sie auch deren Pfad mit angeben.

Nicht-HTML-Dokumente erzeugen

Grundsätzlich ist Virthos dafür gedacht, dynamische Webseiten zu erzeugen, die durch HTML-Templates definiert werden. Virthos Master geht in diesem Punkt noch einen Schritt weiter und erlaubt es, Templates in anderen Formaten bereitzustellen, um damit beispielsweise dynamische Word- oder Excel-Dokumente zu erzeugen. Grundsätzlich sind alle Dateiformate möglich, die auf reinem Text basieren und keine Binärdaten enthalten, sprich: die sich mit einem einfachen Texteditor öffnen lassen (auch wenn der Inhalt einem menschlichen Leser vielleicht nicht unbedingt viel sagt).

Nehmen wir an, Sie möchten eine Excel-Tabelle mit einer Liste aller Seiten erzeugen, die sich innerhalb des Startseitenzweiges befinden. Erstellen Sie dazu in einem Texteditor eine Datei mit folgendem Inhalt:

```
Nummer|erstellt am|Name {{vtSelect: -origin="/"}}{{vtLoop}}
{{vtID}}|{{vtCreationDate}}|{{vtName:raw}}{{vtEndLoop}}{{vtEndSelect}}
```

Ersetzen Sie dabei senkrechten Striche (|) durch Tabulatorzeichen.

Speichern Sie die Datei unter dem Namen *seitenliste.xls* in dem Verzeichnis, in dem sich auch die übrigen Templates des Webauftritts befinden. Rufen Sie anschließend die folgende URL in Ihrem Webbrowser auf:

```
http://.../virthos.php?-met=seitenliste.xls
```

Nach einer mehr oder weniger langen Wartezeit (je nach Größe des Webauftritts) beginnt der Webbrowser mit dem Herunterladen der Excel-Datei. Wenn Sie den Internet Explorer verwenden, stellt dieser den Inhalt der Datei möglicherweise sofort dar. Ansonsten können Sie die Datei mit einem Doppelklick in Excel öffnen, um sich die Seitenliste anzuschauen.

Damit der Webbrowser weiß, wie er mit einer Nicht-HTML-Datei, die Sie bereitstellen, umgehen soll, müssen Sie beim Benennen dieser Datei auf die richtige Namensendung achten. Virthos kann viele Endungen automatisch in den richtigen MIME-Typ umsetzen. Die folgende Tabelle zeigt, welche dies sind:

Endung	MIME-Typ
css	text/css
csv	text/comma-separated-values
doc	application/msword

Endung	MIME-Typ
htm, html	text/html
js	text/javascript
pdf	application/pdf
rtf	application/rtf
sgm, sgml	text/x-sgml
txt	text/plain
xfdf	application/vnd.adobe.xfdf
xls	application/msexcel
xml	text/xml

Wichtig: Wenn Sie einen freien Platzhalter in einem Nicht-HTML-Template verwenden, sollten Sie diesen in der Regel mit dem Zusatz "raw" versehen, wie im obigen Beispiel beim vtName-Platzhalter geschehen. Andernfalls würden Umlaute und Sonderzeichen in HTML-Entitäten (z. B. `&auml;`) umgesetzt, die meistens nicht erwünscht sind. Wenn Sie sicher sind, dass ein Platzhalter keine solche Zeichen enthält (wie im obigen Beispiel bei der Seitennummer oder dem Erstellungsdatum der Fall) können Sie diesen Zusatz auch weglassen.

Hinweise zum Erzeugen von Excel-Tabellen

Das Excel-Dateiformat ist eigentlich ein binäres Format und somit für die Verarbeitung durch Virthos nicht geeignet. Das obige Beispiel zeigt aber, dass man sich behelfen kann. Excel interpretiert nämlich auch reine Textdateien, die mit der Endung *.xls* versehen sind, korrekt. Umlaute und Sonderzeichen werden in diesem Fall jedoch nur unter Windows korrekt dargestellt, weil Virthos diese Zeichen im ISO-Latin1-Zeichensatz codiert, der weitenteils identisch ist mit dem Windows-Zeichensatz. Die Mac-Version von Excel nimmt dagegen an, dass eine reine Textdatei im Mac-Zeichensatz vorliegt, und stellt die Zeichen somit falsch dar.

Eine Excel-Tabelle als reine Textdatei bereitzustellen, hat noch einen anderen Nachteil: Man kann keine unterschiedlichen Schriftgrößen, -farben und -formatierungen (fett, kursiv etc.) verwenden. Um diesen Nachteil zu umgehen, kann man eine Excel-Tabelle auch im HTML-Format bereitstellen, wiederum mit der Endung *.xls*. Um herauszufinden, wie eine solche HTML-Datei aufgebaut sein muss, speichern Sie am besten eine bestehende Excel-Datei im HTML-Format ab und analysieren den Quellcode, der dabei entsteht. Wichtig ist, dass Sie bei freien Platzhaltern den "raw"-Zusatz weglassen, denn in diesem Fall ist die Umwandlung von Sonderzeichen in Entitäten ja erwünscht.

Schließlich sei auch noch auf die Möglichkeit hingewiesen, Excel-Tabellen als XML-Dateien bereitzustellen. Die neueren Excel-Versionen können Tabellen im XML-Format lesen und speichern, und im Unterschied zu HTML bleiben dabei *alle* Aspekte der Tabellen erhalten, beispielsweise auch mehrere Arbeitsblätter. Allerdings ist das von Excel verwendete XML-Format sehr komplex, und das Erstellen eines Templates somit aufwendig.

Hinweise zum Erzeugen von Word-Dokumenten

Auch das Word-Dateiformat ist ein binäres Format und somit für die Verarbeitung als Virthos-Template nicht geeignet. Aber auch hier kann man sich behelfen, denn Word unterstützt – wie viele andere Schreibprogramme auch – das Rich-Text-Format (RTF), das auf reinem ASCII-Text basiert. Selbst Bilder werden durch Textzeichen codiert, was zwar die Dateigröße anwachsen lässt, aber die Verwendung für Virthos-Templates einfach macht.

Im Quellcode ist eine RTF-Datei nicht sehr gut les- und bearbeitbar, denn dieses Dateiformat ist für die maschinelle Verarbeitung gedacht, nicht für die Programmierung von Hand. Aus diesem Grund verfügt Virthos Master über einen speziellen RTF-Modus, der es möglich macht, dass man Word selbst als Template-Editor verwendet: Nehmen Sie ein beliebiges Word-Dokument, fügen Sie ein paar Virthos-Platzhalter (wie üblich in doppelten geschweiften Klammern) ein, und speichern Sie es als RTF-Datei – fertig ist das RTF-Template. Die geschweiften Klammern werden zwar im RTF-Quellcode anders codiert als üblich, aber wenn ein Template mit der Endung *.rtf* abgespeichert ist, berücksichtigt Virthos dies automatisch.

Leider funktioniert die Sache aber nicht immer so problemlos, denn manchmal fügt Word mitten in einem Virthos-Platzhalter einen Zeilenumbruch oder irgendwelche Formatierungszeichen ein. In Word selbst sieht man nichts davon, aber im RTF-Quellcode – und nur dieser ist für Virthos sichtbar – stimmt die Syntax nicht mehr. Die Folge ist, dass man nach der Verarbeitung in Virthos nicht das gewünschte Ergebnis sieht und die erzeugte RTF-Datei möglicherweise überhaupt nicht mehr geöffnet werden kann.

In einem solchen Fall muss man sich dann doch mit dem RTF-Quellcode beschäftigen und versuchen, die zerstörten Platzhalter zu finden und wiederherzustellen. Zeilenumbrüche innerhalb eines Platzhalters kann man einfach löschen, da sie nur aus historischen Gründen (weil ältere Computersysteme keine überlangen Zeilen verarbeiten konnten) vorhanden sind. Die *echten* Zeilenumbrüche, die im Dokument vorkommen, werden ohnehin durch spezielle Befehlszeichen oder -wörter dargestellt. Wenn man mitten in einem Platzhalter kryptische RTF-Codes findet, darf man diese nicht einfach löschen, weil die RTF-Datei sonst in der Regel unbrauchbar wird. Stattdessen sollte man sie vor oder hinter den Platzhalter verschieben.

Beachten Sie, dass geschweifte Klammern in der RTF-Syntax eine besondere Bedeutung haben. Sie dienen dazu, Codeblöcke voneinander abzugrenzen. Damit die geschweiften Klammern, die man in Word eingibt, den RTF-Code nicht durcheinanderbringen, wird diesen im

Quellcode automatisch ein umgekehrter Schrägstrich ("Backslash") vorangestellt. Ein typischer Virthos-Platzhalter sieht dann so aus:

```
\{\{Headline\}\}
```

Bei RTF-Templates, und nur bei diesen, akzeptiert Virthos diese Platzhalter-Schreibweise. Die herkömmliche Schreibweise ohne Schrägstriche würde dagegen nicht nur Word durcheinanderbringen, sondern auch in Virthos nicht funktionieren.

Schließlich noch ein Hinweis zum Umgang mit Sonderzeichen: Der RTF-Standard unterstützt verschiedene Arten der Zeichencodierung, aber leider hat man als Anwender in den seltensten Fällen Einfluss darauf. Wenn Sie mit Word ein RTF-Template erstellen, wird dieses unter Windows in einer anderen Codierung gespeichert als unter Mac OS X. Virthos unterstützt zur Zeit nur die Windows-Codierung, deshalb lässt sich die Mac-Version von Word nicht als Template-Editor verwenden. Gleiches gilt für die meisten anderen Schreibprogramme auf dem Mac. Allerdings werden die unter Windows erzeugten Templates auch auf dem Mac korrekt dargestellt.

Cookies setzen und auslesen

Virthos Master erlaubt es, Cookies auszulesen, die im Webbrowser eines Besuchers gespeichert sind. Dies geschieht mit Hilfe des Platzhalters `vtCookie`:

```
<!--{{vtIf: {vtCookie: lastVisit} .neq. }}
    <p>Sie waren zuletzt am {{vtCookie: lastVisit}} bei uns.</p>
<!--{{vtElse}}-->
    <p>Sie sind zum ersten Mal hier!</p>
<!--{{vtEndIf}}-->
```

Um Cookies zu setzen, steht in Virthos Master eine eigene Aktion namens "setCookie" zur Verfügung, die in der Regel als Inline-Aktion verwendet wird:

```
<!--{{vtDo:-act="setCookie", lastVisit="{vtCurrentDate}", -expire="3600"}}-->
```

Der -expire-Parameter gibt an, wie lange (in Sekunden) der Cookie gespeichert bleiben soll. In dem Beispiel wäre es eine Stunde. Lässt man den Parameter weg, bleibt der Cookie für 30 Tage gespeichert.

Daneben stehen weitere Parameter zur Verfügung, um die Cookies auf eine bestimmte Domain oder einen bestimmten Pfad zu beschränken:

```
<!--{{vtDo:-act="setCookie", cookie="Wert",
            -domain="www.virthos.net", -path="/partner/"}}-->
```

Statistische Auswertungen

Virthos Master stellt einen zusätzlichen Platzhalter namens "vtEval" zur Verfügung, mit dem sich innerhalb einer vtSelect-Blocks einfache statistische Auswertungen durchführen lassen. Ein Beispiel könnte so aussehen:

```
<!--{{vtSelect: -type="person", -filter="{Groesse}.gt.0" }}-->
    <p>Die größte Person ist {{vtEval: max, Groesse }} cm groß.</p>
<!--{{vtEndSelect}}-->
```

In diesem Fall würde der vtEval-Platzhalter durch den höchsten Wert ersetzt, der bei den Seiten vom Typ "person" im Feld "Größe" gespeichert ist.

Zur Zeit stehen die folgenden Auswertungen zur Verfügung:

Schreibweise	Ergebnis
{{vtEval: count}}	Anzahl der selektierten Datensätze (oder Felder mit Inhalt???)
{{vtEval: sum, Feldname}}	Summe der Feldinhalte
{{vtEval: min, Feldname}}	Minimum der Feldinhalte
{{vtEval: max, Feldname}}	Maximum der Feldinhalte

Damit eine Auswertung richtige Ergebnisse liefert, dürfen in dem verwendeten Platzhalter nur reine Zahlenwerte stehen. Es empfiehlt sich daher, wie im obigen Beispiel, stets eine Filterbedingung zu definieren, die einen numerischen Vergleich (größer als, kleiner als etc.) durchführt.

Seiten mit Administratorrechten selektieren

Wenn Sie mit vtSelect und vtLoop eine Seitenauswahl definieren oder durchlaufen, werden dabei im Normalfall nur Seiten berücksichtigt, für die der aktuelle Benutzer ein "Sehen"-Recht besitzt. Manchmal ist es jedoch wünschenswert, diese Beschränkung aufzuheben und allen Benutzern die gleiche (vollständige) Auswahl zu zeigen. Virthos Master stellt für diesen Fall eine spezielle Anweisung zur Verfügung, mit der nachfolgende Anweisungen so ausgeführt werden, als besäße der aktuelle Benutzer Administratorrechte. Das folgende Beispiel zeigt, wie das aussehen kann:

```
<!--{{vtAsAdmin}}-->
<!--{{vtSelect: -origin="///Benutzer/Redaktion", -type="vtuser"}}-->
    <select name="Redakteur">
    <!--{{vtLoop}}-->
        <option value="{vtID}">{{vtName}}</option>
    <!--{{vtEndLoop}}-->
    </select>
<!--{{vtEndSelect}}-->
<!--{{vtEndAsAdmin}}-->
```

Hiermit wird eine Auswahlliste angezeigt, in der alle Benutzer aufgeführt sind, die zur Gruppe "Redaktion" gehören. Ohne die `vtAsAdmin`-Anweisung wäre die Liste leer, wenn der aktuelle Benutzer keine Administratorrechte besitzt, denn die Seiten, die zur Virthos-Benutzerverwaltung gehören, sind nur für Administratoren zugänglich. Dank der `vtAsAdmin`-Anweisung erhält jeder Benutzer das gewünschte Ergebnis.