KARRIEREhandbuch.php5 – Projektdokumentation

Markus-Hermann Koch
mhk@karrierehandbuch.de

11. Februar 2010

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Projektbeschreibung

1.1 Allgemeines

Die Neufassung des Karrierehandbuchs lässt sich prinzipiell in zwei Teile untergliedern:

1. Skripte außerhalb der Datenbank. Diese stellen Darstellungsstile ein, verteilen Inhalte und klären Berechtigungsfragen.

2. Die Datenbank `karriere`. Sie enthält die Strukturtabellen sowie sämtliche Text-Inhalte der Site (die Ausnahme bilden kontrollierte Abstürze begleitende Fehlermeldungen), Kundendaten, Benutzerdaten, Alias – Preferences, die online-Tabelle und noch einiges mehr.

1.2 Außerhalb der Datenbank

1.2.1 Formatierung mit CSS

Das Front End `karrierehandbuch.php5` und die Hauptseite des Back Ends `fgMain.php5` sind im Grunde HTML-Vorlagen die mit Hilfe einer Reihe von objektorientierten PHP-Skripten Datenbankinhalte formatieren und auf eine Folge von `<div id='..'>`- Umgebungen verteilen. Abbildung 1 zeigt das Div Setting beider Vorlagen. Die inneren Div's werden innerhalb von

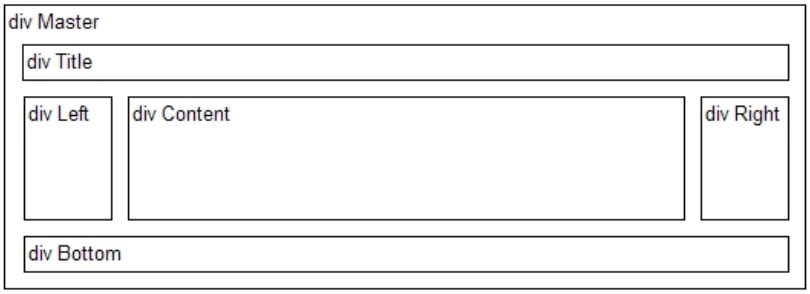

Abbildung 1: Aufbau der Hauptvorlagen

Div Master absolut positioniert. Div Master selbst ist 1000px breit und hat `min-height: 550px`, wobei das Java-Script `vScale.js` diesen Wert ggf. auf `clientHeight` setzt.

Das Div **Title** enthält lediglich die 1000px breite und 95px hohe Titelbildgrafik.

Die Divs **Left** und **Right** nutzen `staticTools::linkList` um aus den Strukturtabellen `index_kh` bzw. `index_redaktion` die Hauptlinkleisten zu generieren. Der Impressumsblock ist ein statischer Inhalt aus `statcon_<lng>`.

Das Div **Content** ist als `overflow: auto` deklariert und wird mit einem Inhalt aus der Datenbank befüllt der vorher durch ein Objekt der Klasse `<cconv>Disp` formatiert wurde.

Das Div **Bottom** schließlich enthält die Fußnote. Auch sie ist ein statischer Inhalt aus `statcon_<lng>`.

Da die Seiten XHTML 1.0-kompatibel gesetzt sind werden nur elementare HTML-Tags verwendet und fast die gesamte Formatierung ist in `.css`-Dateien ausgelagert:

`common.css`	Wird immer zuerst eingebunden. Häufig auftretende Umgebungen erhalten hier erste Stildefinitionen. Diese werden bei Bedarf von nachfolgenden Style Sheets überschrieben.
`main.css`	Hauptstylesheet für `karrierehandbuch.php5`. Hier werden die oben genannten Haupt-Div-Umgebungen gesetzt.
`fgMain.css`	Hauptstylesheet für `fgMain.php5`.
`dbKonsistenz.css`	Hauptstylesheet für `dbKonsistenz.php5`.
`redaktion.css`	Hauptstylesheet für `redaktion.php5`.
`<cconv>Con.css`	Zusätzliches Stylesheet für Inhalte die als Content Converter die Klasse `<cconv>Disp` benutzen.
`<cconv>Toc.css`	Zusätzliches Stylesheet für Inhaltsverzeichnisse die als Content Converter die Klasse `<cconv>Disp` benutzen.

1.2.2 Grobe Übersicht über den Aufbau der PHP-Struktur

Das Rückgrat des Programmes bilden die PHP-Skripte. Abbildung 2 gibt einen Überblick über die Abläufe beim Aufbau einer Hauptseite.
Zu Beginn werden die Klasse `staticTools`, das Skript `kinit.php5` und je nach Einstellung der dort definierten globalen Variablen `chooseInit` ein weiteres, plattformabhängiges Initialisierungsskript eingebunden.

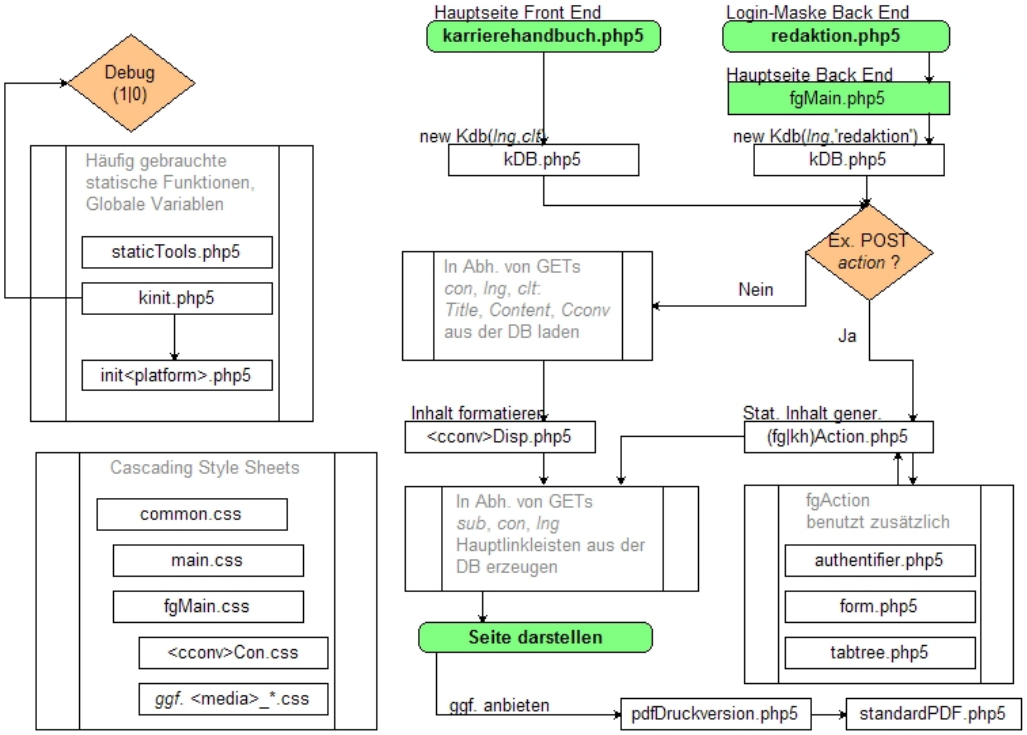

Abbildung 2: Flussdiagramm zum Hauptseitenaufbau

Die Style Sheets werden nach Bedarf in der angegebenen Reihenfolge eingebunden. Es folgt eine Kurzbeschreibung der beteiligten Skripte.

- `kinit.php5`: PHP-Skript. Hier werden alle über PHP gesteuerten Einstellungen vorgenommen. Per Konvention beginnen alle hier deklarierten und auch außerhalb von `kinit` verwendeten Variablen mit der Zeichenfolge `i_`. Besondere Erwähnung verdient die boolsche Variable `i_debug`. Falls `true` läuft die Seite im Debug-Modus. Das sorgt dafür, dass das Programm auch bei abfangbaren Fehlern kontrolliert mit einer im Quellcode verankerten Fehlermeldung abstürzt. Im Online-Betrieb sollte `i_debug=false` gesetzt werden.

- `init<platform>.php5`: Plattformabhängiges, zusätzliches Initialisierungsskript. Die genaue Auswahl wird in `kinit.php5` über die Variable `chooseInit` festgelegt. `init<platform>.php5` enthält die Zugangscodes für Datenbank und Server sowie die Initialisierung der Stammpfade `i_html` für Browser-Links und `i_root` für festplattenbasierte Pfadangaben.

- `staticTools`: Eine Klasse statischer Werkzeuge die „immer mal wieder“ für verschiedene Zwecke benötigt werden.

- `kDB.php5`: Enthält die umfangreiche Klasse `Kdb`. Diese steuert sämtliche Kontakte mit der Datenbank.

- `fgAction.php5`: Enthält die gleichnamige Klasse. Diese wird von `fgMain.php5` benötigt falls an `fgMain.php5` ein Parameter `action` gesandt wird (per POST oder per GET). Falls dies der Fall ist wird `fgMain` instanziert und der Konstruktor des Objektes ruft die Methode mit dem von `action` angegebenen Namen auf. Auf diese Weise funktionieren fast alle Formularseiten der Redaktion.

- `authentifier.php5`: Diese von den Redaktions-Teilen des Projektes benötigte Klasse ist für das Ein- und Ausloggen von Benutzern sowie das Klären von Rechtefragen zuständig. Sämtliche Cookies betreffende Methoden sind hier untergebracht.

- `baum.php5`: Dieser Klasse wird im Wesentlichen eine Index-id, ein Sprachkürzel und ein Client-Suffix übergeben. Sie analysiert dann die Umgebung in der Baumstruktur um den durch die id angegebenen Knoten und zeichnet eine einfache png-Grafik mit diesem Knoten in der Mitte und den Parents und Kids auf der linken und auf der rechten Seite. Außerdem liefert die zentrale Methode `createTree` eine HTML-Area-Map zurück die die einzelnen auf der Grafik dargestellten Felder in Links umwandelt. Die Grafik wird im `/tmp`-Verzeichnis abgelegt. Die meisten Einstellungen für die Skalierung der Grafik befinden sich in `kinit.php5`.

- `htDisplay.php5`: Basisklasse für alle Darstellungsklassen. Enthält unter anderem die Methode `replaceAliases` der das Kapitel 2 gewidmet ist. `htDisplay` enthält außerdem ein Interface für die Methoden `conTpl`, `tocTpl` und `placeAd` die in `htDisplay` selbst nur sehr rudimentär implementiert sind. Ausgefeilte Versionen dieser Methoden befinden sich in den von `htDisplay` abgeleiteten Klassen. In der Basisversion sinnvoll anwendbar ist lediglich `conTpl()`: Darzustellende Contents werden durch `replaceAliases` geschickt und danach direkt an den Browser weitergeleitet.

- `<cconv>Disp.php5`: In den Einträgen der Content-Tabellen gibt es auch ein Feld `cconv` in dem angegeben werden kann welcher Art der zugehörige Inhalt ist. Für den normalen Karrierehandbuch-Artikel und die gewöhnlichen Checklisten wird hier `standard` angegeben. Sollte `cconv` nicht leer sein wird der Inhalt nicht von `htDisplay` sondern von der abgeleiteten Klasse `<cconv>Disp` formatiert bevor er angezeigt wird. `<cconv>Disp` erbt alle Methoden von `htDisplay`, implementiert aber die Formatierungsmethoden `conTpl`, `tocTpl` und `placeAd` den Anforderungen entsprechend neu. Auf diese Weise können Autoren relativ einfach strukturierte Artikel schreiben die dann von der gewählten `htDisplay`-Ableitung auf die gewünschte Form gebracht wird.

1.3 Innerhalb der Datenbank

Dieses Unterkapitel bezieht sich auf die mySQL-Datenbank `karriere`.

Im Folgenden werden die einzelnen Felder der Tabelle auch als Spalten, die einzelnen Datensätze auch als Zeilen bezeichnet.

1.3.1 Allgemeines

- Die einzige auf die Datenbank zugreifende Klasse ist `Kdb`.
- Jede mySQL-Tabelle verfügt über eine Reihe von Spalten, darunter auch eine die als „Primary Key“ erklärt ist. Die Einträge dieser Spalte müssen paarweise verschieden sein. Außerdem wird in der `karriere`-DB die Konvention eingehalten, dass der Primary Key in der ersten Spalte erfasst ist.
- Mit Ausnahme der Tabelle `online`, in der diese Rolle von der Primary id `id` übernommen wird, enthalten alle Tabellen als zweite Spalte das Merkmal `timestamp` vom Typ `timestamp` mit dem Attribut `ON UPDATE CURRENT_TIMESTAMP`
- Sämtliche Tabellen enthalten als letzte Spalte das Merkmal `comment` vom Typ `text`. Dieses soll keinem programmrelevanten Zweck dienen sondern lediglich eine Möglichkeit zur Kommentierung innerhalb der Datenbank bieten.
- Ein Merkmal heißt *zeilenweise organisiert*, wenn das Programm bei der Interpretation davon ausgeht, dass ein Zeilenumbruch innerhalb des Eintrages auch einem Zeilenumbruch in der Ausgabe entsprechen soll.
- Die Basisnamen (alles vor dem ersten Underscore) der Tabellen sind in `kinit.php5` in den Variablen `i_sql_<Basisname>` erfasst.
- Die Spaltennamen der einzelnen Tabellen sind in `kinit.php5` in den Arrays `i_sql_<Basisname>_keys` erfasst.
- Eine weitere in `kinit.php5` erfasste Kenngröße sind die Hash-Tables `i_sql_<Basisname>_extendable`. Diese haben als Schlüssel das Array `i_sql_<Basisname>_keys` und als Werte
  - 0, falls Modifikation eines Spalteneintrages eine komplette Verwerfung des bisherigen Spalteneintrages bedeutet.
  - 1, falls es unter Umständen sinnvoll sein kann den bestehenden Eintrag zu erweitern. Dies trifft auf Texte und Listen zu.

  - einen String `$s` mit einer Trennzeichenfolge, falls die Spalte eine Liste enthält. Diese Liste ist technisch gesehen ein String der über das PHP-Kommando `explode($s,..)` in ein Array eingelesen werden kann. Die meisten Listen verwenden `$s==';'` als Trennzeichen. Fürderhin wird von einer `$s`-*Liste* gesprochen wenn ein durch `$s` unterteilter Listenstring gemeint ist.

- Der verwendete Zeichensatz für alle Inhalte ist UTF-8.

1.3.2 Struktur der Tabelle `client`

Die Site ist in Mandanten (intern „Clients“) eingeteilt. Sämtliche Clients werden in der Tabelle `client` erfasst.

- `(varchar)cltsuffix`: Das eindeutige Handle des Clients. Das Karrierehandbuch selbst ist der der Primärclient mit dem Kürzel `kh`. Das Kürzel wird verwendet, um den Client innerhalb des Programmes zu identifizieren. Es dient unter anderem bei der Namensgebung der Clientabhängigen Tabellen, bei der Identifizierung über den GET-Parameter `clt` und bei der Klärung von berechtigungsfragen über die Tabelle `authority`.

- `(varchar)client`: Name des Mandanten wie er dargestellt werden soll, wenn er als Inhalt auftaucht.

- `(varchar)lng`: Sprachkürzel nach ISO 639-1. Primärsprache des Mandanten. Inhalte werden per Default in dieser Sprache dargestellt. Sollten Inhalte in einer vom Leser gewünschten Sprache nicht verfügbar sein wird automatisch versucht den Inhalt in der Defaultsprache darzustellen.

- `(varchar)tel,fax,email,(text)home,address`: Persönliche Daten des Kunden wie sie ggf. auch als Inhalte dargestellt werden sollen. Die letzteren beiden sind zeilenweise organisiert und erfordern per se keine weiteren HTML-Tags.

1.3.3 Struktur der Tabellen `index_<clt>`

Jedem Client ist eine Index-Tabelle `index_<clt>` zugeordnet. Die einzelnen Zeilen der Indextabelle bilden die Knoten einer Baumstruktur an die die einzelnen sprachabhängigen Inhalte aus den weiter unten beschriebenen `content`-Tabellen angehängt werden.

- `(int)id`: Der Primärschlüssel. Der Wurzelknoten hat `id=1`.

- `(text)pid`: ;-Liste aus Integerwerten. Jeder einzelne stellt die `id` eines Knoten dar, der dem Knoten als „parent“ zugeordnet sein soll. Im Regelfall ist das nur ein einziger. Das Programm unterstützt aber auch die Haltung mehrerer Parent-Knoten. Jeder Knoten benötigt jedoch mindestens eine `pid`. Die einzige Ausnahme hiervon bildet der Wurzelknoten mit der `id==1`

- `(text)kid`: ;-Liste aus Integerwerten. Jeder einzelne stellt die `id` eines Knoten dar, der dem Knoten als „kid“ zugeordnet ist. Hier ist eine gewisse Redundanz im Konzept: Eindeutigkeit wäre allein durch die Angabe der `pid` bereits gegeben. Sie wird aus Gründen der Übersichtlichkeit und aus Performance-Gründen in Kauf genommen. Überprüfung der Index-Tabelle auf Widersprüche an dieser Stelle ist eine der Hauptaufgaben von `dbKonsistenz.php5`.

- `(int)tid`: Manchen Artikeln im Karrierehandbuch ist ein themenfremder Hinweis der Art „Klicken Sie auf die Überschrift um...“ beigefügt. Diese Hinweise werden in eigenen Sub-Contenttabellen `content_<lng>_<clt>_tip` abgelegt und erhalten dort eine eigene `id`. Da diese Tipps wie gesagt nichts mit dem Inhalt an sich zu tun haben, wurden sie in die Index-Struktur aufgenommen. Die Angabe einer `tid` wird den dargestellten Inhalt um den zugehörigen Tipp erweitern, falls letzterer definiert ist.

- `(int)cid`: Die Content-id des dem Knoten zugeordneten Inhalts. Das Programm sucht dann in der passenden Tabelle `content_<lng>_<clt>` denjenigen Eintrag heraus der die `content_<lng>_<clt>.id` vom Wert `index_<clt>.cid` trägt.
  Wo es möglich ist sollte `index_<clt>.id==index_<clt>.cid` gewählt werden.

- `(text)ctab`: Veraltete Spalte. War dazu gedacht anzugeben welchen Basisnamen die Inhalts-Tabelle hat. Wurde durch das Suffixsystem des Mandantenkonzeptes abgelöst: Alle Inhalte stammen nun aus Tabellen mit dem Basisnamen `content`.

- `(text)section`: Unreife Spalte. Im Augenblick kommt das Rechtesystem der Redaktion mit einigen Grundrechten aus. Es ist jedoch angedacht Benutzern eines Tages Spezialrechte für bestimmte Bereiche (Advertisement, Stellenangebote, etc.) zu geben. In die Spalte Section könnte dann der Name eines solchen Bereiches eingetragen werden und nur Benutzer mit globalen Schreibrechten oder aber dem zusätzlichen Recht `s[<section>]` können dann darauf zugreifen. Um diese Idee umzusetzen müssen einige relativ einfache Erweiterungen in `authentifier.php5` vorgenommen werden.

1.3.4 Struktur der Tabellen `content_<lng>_<clt>`

- `(int)id`: Primärschlüssel. Wird von den `index_<clt>.cid` referenziert.
- `(text)title`: Titeltext für den Artikel.
- `(mediumtext)content`: Eigentlicher Inhalt des Artikels. Sollte keiner angegeben sein wird das Programm versuchen an Stelle eines Inhaltes ein auf dem aktuellen Knoten aufsetzendes Inhaltsverzeichnis zu erzeugen.
- `(text)cconv`: Basisname der Darstellungsklasse `<cconv>Disp` mit deren Hilfe der Inhalt formatiert werden soll. Fall `NULL` wird die Basisklasse `htDisplay` verwendet.
- `(text)returnurl`: Insbesondere bei den mit `standardDisp` formatierten Artikeln ist die Überschrift ein Link auf das nächste übergeordnete Inhaltsverzeichnis. Falls dies einmal nicht gewünscht sein sollte kann hier ein Alternativlink angegeben werden.
- `(int)secretcid`: „Second Return Content-Id“. Manche Artikel, insbesondere Checklisten verfügen über einen anderen Artikel, dem sie direkt zugeordnet sind. Dessen `cid` kann hier angegeben werden. Ist dies der Fall wird in entsprechend formatierten Seiten (bislang implementiert für `standardDisp`) ein zusätzlicher Rückkehrlink (bei `standardDisp` in Gestalt eines kleinen, roten Pfeiles neben der Überschrift) auf diesen Artikel eingeblendet.
- `(bool)toc`: Per Default `NULL`. Falls jedoch ein von 0 verschiedener Wert angegeben wird, interpretiert das CMS den zugehörigen Artikel als Table of Contents. Auch dann, wenn der Inhalt des Artikels nicht leer ist. Das kann benutzt werden, um TOCs von Hand zu erstellen. Eine weitere Hilfestellung zu dieser Aufgabe bietet das Alias `{{CLT...}}` wenn es mit der Option `<title>='toc'` verwendet wird.
  Siehe auch 2.3.26.
- `(text)pdf`: §§-Liste für pdf-Metadaten in der Reihenfolge (Verfasser, Thema, Stichworte). Sollte dieser Eintrag von `NULL` abweichen und der verwendete Stil diese Funktion unterstützen wird dem Artikel ein Kurzformular angehängt mit dessen Hilfe der Betrachter der Seite eine PDF-Datei anfordern kann die dann mit Hilfe des open Source-Paketes `fpdf` aus dem Inhalt heraus erzeugt wird.

1.3.5 Struktur der Tabellen `content_<lng>_<clt>_tip`

- `(int)id`: Primärschlüssel und Handle auf den Tipp.
- `(text)tip`: Text des tipps. Wertet sowohl Aliases als auch HTML aus.

1.3.6 Struktur der Tabellen sa_<lng>_<clt>

Sogenannte „**S**hort **A**dvertisements“ bieten dem Mandanten *A* die Möglichkeit Visitenkarten an beliebige Artikel anzuhängen, sogar auf den Sub-Sites anderer Mandanten. Voraussetzungen hierfür sind

1. ein Eintrag in der Tabelle sa_<lng>_A
2. dass der Zielartikel in einem Stil verfasst ist, der Visitenkarten unterstützt. Bislang ist das nur bei cconv==standard der Fall.

Die Struktur der Tabellen sa_<lng>_<clt> ist wie folgt.

- (int)id: Primärschlüssel und Handle.
- (text)cid: ;-Liste mit den id der Zielartikel. Die Syntax lautet <cid>[@<clt>] und gibt somit die benannte Content Id und optional einen Target Client an. Default für letzteren ist kh.
- (text)content: Zeilenweise organisierter Inhalt für die Visitenkarte. HTML und Aliase werden ausgewertet. Insbesondere die Aliase {{ADD}}, {{EML}}, {{FAX}}, {{HOME}}, {{NAME}} und {{TEL}} sind für Visitenkarten programmiert worden.
- (text)cconv: Angedachter Stilbezeichner für Visitenkarten. Bislang wird lediglich von standardDisp der Stil horizontal unterstützt. Dabei wird der Haupttext der Karte dargestellt und darunter eine Leiste mit Buttons eingeblendet. Diese könnten auch anders platziert werden. Die grundsätzliche Möglichkeit wird durch die Anwesenheit von cconv offen gehalten.
- (text)bkimg: Dateiname für eine Hintergrundgrafik der Visitenkarte. Diese soll im Verzeichnis ./Images/<clt> abgelegt sein. Grundsätzlich kann der Dateiname aber auch mit einem Unterpfad ergänzt werden. Die Zeichenfolge '..' wird jedoch gefiltert.
- (text)buttons: \r\n-Liste. Captions für die Buttons der Visitenkarte. Die Anzahl ist lediglich durch die Breite der Visitenkarte und die Länge der einzelnen Captions begrenzt. Aliase werden interpretiert.
- (text)urls: \r\n-Liste. URLs für die Buttons der Visitenkarte. Eine URL die als 0 angegeben ist führt dazu, dass der zugehörige Button nicht angezeigt wird. Eine URL kann sowhl einfach als URL, als auch als Hyperlink oder via geeigneter Aliase angegeben werden.
- (text)section: Nicht implementiert. Analog zum gleichnamigen Merkmal in den Tabellen index_<clt>.

1.3.7 Allgemeines zu statischen Inhalten

Die Inhalte des Karrierehandbuchs bestehen nicht nur aus Artikeln und Inhaltsverzeichnissen. Es gibt auch, insbesondere in der Redaktion, eine ganze Reihe von Textinhalten die zwar durchaus sprachflexibel sein sollen, aber unabhängig vom aktiven Kunden dargestellt werden. Der Impressumsblock und die Fußnote sind gute Beispiele für solche Inhalte. Aber auch die meisten Formulare aus der Redaktion die sich aus vielen Inhaltsfragmenten zusammensetzen.

Die Tabellen `static` und `statcon_<lng>` wurden mit Blick auf die zuerst genannten „Dauerinhalte“ ins Leben gerufen, die ich damals „statische Inhalte“ taufte. Mittlerweile gibt es für diese statischen Inhalte neben Aliasen noch die weitere Methode über den Platzhalter `%%` und das `staticTool::fmtStatCon()` Parameter zu empfangen. Somit sind sie heute eigentlich sogar die dynamischsten Inhalte des Projektes. Ich bezeichne sie dennoch bis auf weiteres als „statisch“.

Von der Funktionsweise her ist die Tabelle `static` mit den Indextabellen und die Tabellen `statcon_<lng>` sind mit den Contenttabellen verwandt. Ausgelesen werden sie mit Hilfe des statischen Tools `staticTools::fmtStatCon()`.

1.3.8 Struktur der Tabelle `static`

- `(int)id`: Primärschlüssel. Hier ist eine Erklärung notwendig, warum nicht das `handle` über welches jeder statische Inhalt verfügt, diese Funktion übernimmt. Das `handle` ist für sich genommen nicht eindeutig. Das wird es erst zusammen mit dem Merkmal `context`. Auf diese Weise ist es beispielsweise möglich das Handle `title` mehrfach zu vergeben. Dieses ist auch geschehen: Einmal im Context `redaktion` und einmal im Context `fgAction`. Wenn das Programm weiter wächst und weitere htTools hinzustoßen mag sich sogar eines Tages der Wunsch einstellen `title` ein drittes mal zu vergeben.

  Hier wäre es evtl. eleganter gewesen wie bei den Indecies mit mehreren `static`-Tabellen zu arbeiten. Dieses Kurzkonzept war wie gesagt nicht für die Fülle an Aufgaben gedacht gewesen, die es heute erfüllt.

- `(varchar)handle`: Kurzbezeichner für den statischen Inhalt.

- `(varchar)context`: Kontext des statischen Inhaltes. Ein Handle wird nur innerhalb derjenigen Einträge der Tabelle gesucht, die den angegebenen Context haben. Sollte diese suche erfolglos verlaufen wird es das Programm unter dem Context `global` noch einmal versuchen. Inhalte vom Context `global` sind z.B. die KARRIERE-Fußnote und der Impressumsblock.

- `(int)cid`: Content id. Entspricht der `statcon_<lng>.id` der der eigentliche Inhalt entnommen wird. Wo es möglich ist, sollte `index_<clt>.id==index_<clt>.cid` gewählt werden.

1.3.9 Struktur der Tabellen `statcon_<lng>`

- `(int)id`: Primärschlüssel. Wird über `static.cid` angesteuert.
- `(text)content`: Der eigentliche Inhalt. Kann wie eine HTML-Seite programmiert werden. Kann Aliase enthalten. Kann Platzhalter `%%` für über `staticTools::fmtStatCon()` übergebene Parameter enthalten. Die Erfahrung lehrt, dass statische Inhalte eher kurz sein sollten. Ein kurzer Inhalt kann an zweiter Stelle leichter wiederverwendet werden als ein zu ausführlich geratener.

1.3.10 Struktur der Tabellen `download_<lng>_<clt>`

Diese Tabellenklasse wurde eingens für das Alias `{{DWN..}}` eingeführt und enthält alle Daten, um einen Link auf einen im Verzeichnis `~/Downloads/<clt>` bereitliegenden Download zu erzeugen.

- `(ind)id`: Primärschlüssel und Handle.
- `(text)title`: Für die gleichnamige Eigenschaft im HTML-Anchor.
- `(text)subdir`: Unterverzeichnis der Zieldatei. Das Programm geht davon aus, dass der Download im Stammverzeichnis `~/Downloads/<clt>` liegt.
- `(text)fname`: Dateiname des Zieldatei. `$subdir.'/'.fname` wird durch eine Sicherheits-RegExp gefiltert.
- `(text)section`: Nicht implementiert. Analog zum gleichnamigen Merkmal in den Tabellen `index_<clt>`.

1.3.11 Struktur der Tabelle `aliases`

`aliases` enthält die Handles der Aliases, Parameter für einige der einfacheren Exemplare, sowie eine während der Programmierung getippte Kurzbeschreibung. Die einzige Methode die auf diese Tabelle zugreift ist `htDisplay->internal_evalAlias`. Sie ist das Arbeitspferd von `htDisplay->replaceAliases` und implementiert sämtliche Aliases.

Zur Tabelle `karriere.aliases`.

- `(varchar)alias`:
- `(text)preferences`: Den Aliasen kann hier eine `%%`-Liste an Parametern übergeben werden. Diese kommen in `htDisplay->internal_evalAlias` als Array `$pref` an.

- `(text)man`: Kurz-Beschreibung des Aliases durch den Programmierer. Sollte auf die Syntax, die Bedeutung der Parameter und die Ausgabe eingehen.

1.3.12 Struktur der Tabelle `authority`

Mit Einführung der Redaktion als Mandantenfähiges Back End wurde die Verwaltung von Benutzern mit unterschiedlich gestalteten Zugriffsrechten erforderlich.

Diese User werden `authority` verwaltet.

- `(varchar)login`: Primärschlüssel, Login und Handle des Benutzers. Zur generierung des Login-Namens wird die Methode des Stud-Pools des mathematischen Fachbereichs an der Uni Stuttgart imitiert: Das Login setzt sich zusammen aus den ersten sechs (soweit vorhanden) Buchstaben des Nach- sowie dem ersten und letzten Buchstaben des Vornamens des Benutzers. Alles klein geschrieben. Gustav Gans hätte folglich das Login `gansgv`.
- `(varchar)address`: Die Anrede des Benutzers. Z.B. „Frau“, „Herr“, „Meister“, etc.
- `(varchar)firstname`: Der Vorname des Benutzers.
- `(varchar)lastname`: Der Familienname des Benutzers.
- `(varchar)lng`: Nicht implementiert. Die bevorzugte Sprache des Benutzers. Kürzel nach ISO 639-1.
- `(varchar)email`: Die E-Mail-Adresse des Benutzers.
- `(varchar)passwd`: Das Passwort des Benutzers. Um die Privatsphäre der Benutzer an dieser Stelle etwas zu verstärken werden die Passwörter vor dem Eintrag in die Datenbank über das statische Tool `Kdb->encrypt($msg,$key,$secretize)` verschlüsselt.
- `(varchar)cltsuffix`: `\r\n`-Liste. Enthält Client-Suffixe jener Mandanten auf die der betreffende User Zugriff hat. Für die Hauptadmins der Site ist der spezielle Mandant `anyclient` eingeführt worden. Falls `anyclient` angegeben wird darf die Liste sonst keine weiteren Mandanten mehr enthalten.
- `(varchar)auth`: `\r\n`-Liste der Rechtestrings. Korrespondiert mit der Liste `cltsuffix`: Dem j. Element der `cltsuffix`-Liste wird der j. Rechtestring der `auth`-Liste zugeordnet.
- `(int)lastvisit`: Nicht implementiert. Unix-Timestamp des letzten Logins.

- `(int)expires`: Unix Timestamp des Account-Ablaufs. Nachdem ein Account abgelaufen ist kann sich der betroffene Benutzer nicht mehr einloggen. Außerdem scheitern alle über die Redaktion geführten schreibenden Zugriffe auf die Datenbank.

1.3.13 Struktur der Tabelle `online`

Diese Relatively High Traffic Tabelle führt Buch über die zu einem bestimmten Zeitpunkt eingeloggten Benutzer. Insbesondere werden die Einträge in dieser Tabelle wieder entfernt wenn

1. sich die Benutzer erneut einloggen oder
2. sich aktiv ausloggen.

Es sei an dieser Stelle auch angemerkt: Um erfolgreich eingeloggt zu sein ist es notwendig in der Online-Tabelle einen entsprechenden Eintrag zu halten. Hinreichend ist es nicht.

- `(int)id`: Primärschlüssel. Verwendet wird der Unix-Timestamp des Einlogvorgangs. Sollte diese `id` durch einen ungeheuren Zufall jedoch bereits vergeben sein, wird so oft 1 addiert bis eine freie `id` gefunden wurde.
- `(varchar)login`: Login des Benutzers.
- `(varchar)ip`: Ip von der aus das Login erfolgt. Das beim einloggen generierte Cookie wird bei einer Anfrage unter anderem nur dann akzeptiert, wenn letztere von der selben ip aus erfolgte wie der Einlogvorgang. Dieses Vorgehen soll Cross Site Scripting erschweren.
- `(text)cookie`: Wert des dem Benutzer beim Einloggen übersandten Cookies.

2 Aliases

2.1 Über Aliase

Ein *Alias* ist eine Zeichenfolge die beim Kompilieren der Seite durch eine andere Zeichenfolge ersetzt wird.

Die Idee mit Aliasen zu arbeiten stammt für mich aus der Zeit an der Uni Stuttgart als Joachim Wipper, Joachim Gaukel und ich unter Anwendung des LaTeX-Paketes `psfrag` das Paket `mtp` programmierten. Mit Hilfe von `mtp` wurde es möglich Matlab-Grafiken so in LaTeX-Dokumente einzubinden, dass sämtliche Beschriftungen der Grafik mit dem von LaTeX verwendeten Font in der im Dokument eingestellten Skalierung neu gesetzt wurden.

Joachim Gaukel entwickelte damals eine ganze Reihe von zusätzlichen Tools für das elegante Erstellen von Grafiken unter Matlab und brachte dabei auch die Idee ein, Aliase als Parameter für die diversen Funktionen zu verwenden. So war es beispielsweise möglich

```
>> mtparrow(..,'red','dotted');
```

in Matlab einzugeben, um einen roten, gepunkteten Pfeil zu erhalten. Der Aliasgedanke und seine Umsetzung trugen einiges zur Benutzerfreundlichkeit von `mtp` bei.

Als ich Anfang 2008 das Karrierehandbuch neu entwickelte und dabei das Ziel verfolgte auch innerhalb der Datenbank Struktur und Inhalt zu trennen stand ich rasch vor der Notwendigkeit, relativ dynamische Strukturinformationen in die an sich recht statischen Inhalte aufzunehmen. So schrieb ich die Tool-Methode `replaceAliases`, deren Aufgabe es war, jeden Text vor der Darstellung im Browser nach Zeichenfolgen der Form

```
%%PHPkaesebrot%%
```

abzusuchen und dann (im Beispiel) durch den Wert der zur Laufzeit im Speicher befindlichen Variablen `$kaesebrot` zu ersetzen. Dies ermöglichte beispielsweise die Aufnahme von wartungsfreien Links in die Artikel des Karrierehandbuchs welche sich automatisch anpassten falls der Artikel auf den sie verwiesen einmal innerhalb der Strukturtabelle umziehen sollte.

Außerdem war der Grundstein gelegt beliebige Zeichenfolgen unter Verwendung des Arbeitsspeichers, der Festplatte und oder der Datenbank zu ersetzen.

Heute ist `replaceAliases` eine der zentralen Methoden der für die Darstellung aller Inhalte verantwortlichen Klassenfamilie `htDisplay` und unterstützt über 60 verschiedene Aliase die bei Bedarf auch ineinander verschachtelt werden können. Die grundsätzliche Syntax hat sich marginal gewandelt:

```
{{<Name des Aliases><Durch ';' getrennte Parameterliste>}}
```

Die Aliase können ganz regulär in Artikeln eingegeben werden. Den Satz „*Wir* sind die **lustigen** Holzhackerbuam !“ könnte man aliasgestützt wie folgt darstellen:

```
{{underline{{emWir}} sind}} die {{bf lustigen}} Holzhackerbuam !
```

2.2 Konventionen

1. Die grundsätzliche Syntax lautet

   ```
   {{<aliasName><';'-getrennte Parameterliste>}}
   ```

   Hierbei kann nur das letzte Argument das Zeichen `;` als Teil seines Inhaltes enthalten.

2. Aliase sind *Case Sensitive*. Unmodifiziert aus LaTeX oder HTML übernommene Aliase sind klein, alle anderen groß geschrieben.

3. Manche ältere Aliase erfordern ein weiteres Alias als Endtag. Dieses hat denselben Namen wie das eröffnende Alias gefolgt von einem kleinen 'e'. Z.B. `{{FORM..}}` und `{{FORMe}}`

4. Jedes Alias ist in der SQL-Tabelle `karriere.aliases` registriert. Dort liegt auch für jedes Alias eine Kurzdokumentation. Außerdem ist dort für die einfacheren Aliase Code hinterlegt. Das Alias `{{smaller..}}` beispielsweise findet dort die Zeichenfolge

   ```
   <span class='smaller'>%%</span>
   ```

   und ersetzt dann nur noch `%%` durch seinen ersten (und in diesem Fall einzigen) Parameterstring.

5. Manche Aliase legen HTML-Umgebungen an. Als `class` wird in der Regel der Name des Aliases angegeben.

6. Eine Sonderrolle nimmt das Alias `{{CHR..}}` ein. Es ist redundant unter `staticTool::hackSecureEncode(..)` implementiert und wird in dieser Form von der Datenbankverwaltungsklasse `Kdb` verwendet. Da `Kdb`-Objekte innerhalb der `htDisplay`-Objekte instanziert werden, muss `Kdb` selbst auf die Funktionalität von `replaceAliases(..)` verzichten. Da `{{CHR..}}` bei der Anti-SQL-Injection-Verschlüsselung zum Einsatz kommt ist es wünschenswert, dass `Kdb` dieses spezielle Alias interpretieren kann.

2.3 Die Aliase

2.3.1 ae

`{{ae}}` wird ersetzt durch `</a>`

2.3.2 bf

„Bold Face.“ `{{bf%%}}` wird ersetzt durch `<span class='bf'>%%</span>`

2.3.3 br

„Line Break.“ `{{br}}` wird ersetzt durch `<br />`

2.3.4 divCenter

`{{divCenter%%}}` wird ersetzt durch `<div class='centerline'>%%</div>`

2.3.5 divLine

`{{divLine%%}}` wird ersetzt durch `<div class='line'>%%</div>`

2.3.6 em

„Emphasis.“ `{{em%%}}` wird ersetzt durch `<span class='em'>%%</span>`

2.3.7 larger

`{{larger%%}}` wird ersetzt durch `<span class='larger'>%%</span>`

2.3.8 smaller

`{{smaller%%}}` wird ersetzt durch `<span class='smaller'>%%</span>`

2.3.9 underline

`{{underline%%}}` wird ersetzt durch `<span class='underline'>%%</span>`

2.3.10 ADD

`\verb|{{ADD[<cltSuffix>[;div]]}}|`

wird ersetzt durch die Adresse des Mandanten mit dem angegebenen Client Suffix. Wird als zweiter Parameter die Zeichenfolge `div` angegeben, so wird die Adresse zusätzlich in `<div class='ADD'>..</div>` verpackt.

Siehe auch: EML, FAX, HOME, NAME, TEL

2.3.11 ADY

`{{AUY}}` ('Arrow Down Yellow') wird ersetzt durch

```
<img class='thumb' src='{{PHPi_protocol}}{{PHPi_html}}
  {{PHPi_images_arrows}}{{PHPi_gif_arrows;dn_yellow}}'
  alt='go to next' />
```

Dies bindet die Grafik eines gelben, nach unten weisenden Pfeils ein.

2.3.12 ALR

`{{ALR}}` ('Arrow Left Red') wird ersetzt durch

```
<img class='thumb' src='{{PHPi_protocol}}{{PHPi_html}}
  {{PHPi_images_arrows}}{{PHPi_gif_arrows;left_red}}'
  alt='siehe auch' />
```

Dadurch wird die Grafik eines roten, nach links weisenden Pfeils eingebunden.

2.3.13 ALLVISITORS

`{{ALLVISITORS}}` wird ersetzt durch die Summe aller Besucherzähler in der Client-Tabelle.

2.3.14 ALY

`{{ALY}}` ('Arrow Left Yellow') wird ersetzt durch

```
<img class='thumb' src='{{PHPi_protocol}}{{PHPi_html}}
  {{PHPi_images_arrows}}{{PHPi_gif_arrows;left_yellow}}'
  alt='see also' />
```

Dadurch wird die Grafik eines gelben, nach links weisenden Pfeils eingebunden.

2.3.15 AN

Opening Tag für einen Anker.

`{{AN[<href>;[<name>]]}}`

- `<href>`: Ziel-Url für den Anker.
- `<name>`: Name für den Anker.

Diese Version des Aliases ist für interne Links gedacht. Der Anker führt nicht dazu, das ein neues Fenster geöffnet wird.

2.3.16 AN*

Variation von `{{AN}}`. Diese Version des Aliases ist für externe Links gedacht und soll eines Tages dazu führen, das bei Aufruf ein neues Fenster geöffnet wird. Dafür kenne ich im Augenblick leider noch keine W3C-XHTML1.0/STRICT standardkompatible Möglichkeit.

2.3.17 ARR

`{{ARR}}` ('Arrow Right Red') wird ersetzt durch

```
<img class='thumb' src='{{PHPi_protocol}}{{PHPi_html}}
  {{PHPi_images_arrows}}{{PHPi_gif_arrows;right_red}}'
  alt='see also' />
```

Dadurch wird die Grafik eines roten, nach rechts weisenden Pfeils eingebunden.

2.3.18 ARY

`{{ARY}}` ('Arrow Right Yellow') wird ersetzt durch

```
<img class='thumb' src='{{PHPi_protocol}}{{PHPi_html}}
  {{PHPi_images_arrows}}{{PHPi_gif_arrows;right_yellow}}'
  alt='see also' />
```

Dadurch wird die Grafik eines gelben, nach rechts weisenden Pfeils eingebunden.

2.3.19 AUY

`{{AUY}}` ('Arrow Up Yellow') wird ersetzt durch

```
<img class='thumb' src='{{PHPi_protocol}}{{PHPi_html}}
  {{PHPi_images_arrows}}{{PHPi_gif_arrows;up_yellow}}'
  alt='go to top' />
```

Dadurch wird die Grafik eines gelben, nach oben weisenden Pfeils eingebunden.

2.3.20 AUTHCLTCOMBO

Alias für eine Combobox die in Abhägigkeit des gesetzten Redaktions-Cookies diejenigen Clients anbietet, auf die der User unter den geforderten Rechten Zugriff hat.

`{{AUTHCLTCOMBO[<reqRights>[;<name>[;<multi>[;<size>]]]]}}`

- `<reqRights>`: Welches Rechtesymbol wird benötigt? Default ist 'a'
- `<name>`: Name für die Combobox. Default ist 'authcltcombo'
- `<multi>`: Boolean. Wird an `{{COMBO..}}` weitergeleitet.
- `<size>`: Wird an `{{COMBO..}}` weitergeleitet.

Auf der technischen Seite erzeugt dieses Alias einen Hash `$authcltcombo` der auf das Array `$GLOBALS` gepusht wird. Danach wird das Alias ersetzt durch

`{{COMBOauthcltcombo;<gefilterte Argumente>}}`

2.3.21 AUY

`{{AUY}}` ('Arrow Up Yellow') wird ersetzt durch

```
<img class='thumb' src='{{PHPi_protocol}}{{PHPi_html}}
  {{PHPi_images_arrows}}{{PHPi_gif_arrows;up_yellow}}'
  alt='go to top' />
```

2.3.22 BBOOK

Ein sehr spezielles, für unser Online-Marketing wichtiges Alias. Bindet ein Thumb-Nail in form eines kleinen, roten Buches ein, welches auf den Artikel `con=29010202,clt=kh` (die Download-Seite des Bewerbungshandbuchs) verlinkt.

Genauer gesagt wird der Code ersetzt durch das in den Alias-Preferences festgelegte Code-Fragment

```
{{CLTkh;29010202}}<img class='thumb' src='{{PHPi_protocol}}
  {{PHPi_html}}{{PHPi_images_arrows}}
  {{PHPi_gif_arrows;book_b_small}}' alt='Bewerbungshandbuch' />{{ae}}
```

2.3.23 BROWSER

Fügt einen grafischen Strukturbrowser der Baum-Klasse ein.
Verpackt ihn in einem `<div class='browser'>..</div>`

```
{{BROWSER<imgSrc>;<imgMap>}}
```

- `<imgSrc>`: Dateiname der aktuellen Browsergrafik wie sie von `Baum->createTree(..)` im Kurzzeit-Tmp-Verzeichnis angelegt wird.
- `<imgMap>`: Area Map-String für die Schaltflächen auf der Grafik wie er von `Baum->createTree(..)` zurückgeliefert wird.

Das Alias wird ersetzt durch

```
<div class='browser'>
<map name='chart'>..</map>
<img alt='Browser'; border='0'; usemap='#chart'; src='..' />
</div>
```

2.3.24 BY

`{{BY}}` „Bull Yellow“. Wird ersetzt durch

```
<span class='gelb'>&bull;</span>
```

und erzeugt ein gelbes Bull's Eye.

2.3.25 CHR

Ersetzt den als Parameter übergebenen Ascii-Code durch das zugehörige Zeichen.

```
{{CHR<(int)ord>}}
```

- `<ord>`: $ord \in \{0, \ldots, 255\}$. Ordnungszahl aus der 8-Bit-ASCII-Tabelle.

Hinweis: Dieses Alias wird redundant von `staticTools::hackSecureEncode(..)` implementiert.

2.3.26 CLT

Aliaspräfix für das Opening Tag eines Anchors auf einen beliebigen Artikel innerhalb der KARRIEREhandbuch-Domain.

```
{{CLT[<(text)cltsuffix>[;<(int)cid>[;<(text)lng>[;<(text)title>
  [;<(bool)plainUrl>]]]]]}}
```

- `<cltsuffix>`: Client-Suffix des Mandanten innerhalb dessen Sub-Site das Ziel liegt. Default ist der aktive Mandant.
- `<cid>`: Content id des Zielartikels. Default ist 1.
- `<lng>`: Kürzel der gewünschten Sprache. Per Default wird die aktuell eingestellte Spreache beibehalten.
- `<title>`:
  - Falls `true`, `t` oder `1` wird die ihrer Tags beraubte Überschrift des Zielartikels als Link Title verwendet. D.h.: Dieses Feature unterstützende Browser zeigen sie an, falls der Betrachter der Seite mit der Maus auf den Link zeigt.
  - Falls `false`, `f` oder `0` ist die Title-Funktion abgeschalten.
  - Falls `toc` wird ein vollständiger Table-Of-Contents-Link erzeugt. D.h.:
    - Es wird kein Title angezeigt.
    - Das Anker-Tag erhält die Eigenschaft `name='<cid>'`.
    - Dem fertigen Opening Tag wird als Link-Text die Überschrift des Zielartikels sowie das Closing-Tag `</a>` mitangehängt.

  Die Defaulteinstellung ist `true`.
- `<plainUrl>`: Falls nichtleer angegeben wird das Alias nur durch die Ziel-Url ersetzt.

2.3.27 COMBO

Alias für eine Combobox.

```
{{COMBO[<(text)hashName>[;<(text name='combo'>
  [;<(bool)multi>[;<(int)size>[;<(bool)selfass>
  [;<(text)selVal>[;<(text)add>]]]]]]]}}
```

- `<hashName>`: Name einer globalen Variablen die einen Hash enthält. Die Schlüssel dieses Hashs stellen die `values`, die Werte die einzelnen `options` der Combobox dar.
- `<name>`: Name für die Combobox. Default ist 'combo'.
- `<multi>`: Boolean. Falls `true` wird die Box als `multichoose` definiert. Sonst nicht. Default ist 0.
- `<size>`: `size` für die Combobox. Wird nicht angegeben falls 0. Default ist 0.
- `<selfAss>`: Boolean. Falls `true` wird der übergebene Hash „selbst-assoziativ". D.h. sich wiederholende Werte werden entfernt und die Anweisung `$hash = array_combine($hash,$hash)` wird ausgeführt. Dies erlaubt die Verwendung eines normalen Arrays dessen Werte in der Combobox sowohl als `values` als auch als `options` dienen.
- `<selval>`: „Selected Value". Kann einen der Schlüssel des die Optionen erzeugenden Hashs enthalten. Dieser Wert wird dann die Voreinstellung der Combo-Box. Per Default ist diese Teilfunktion inaktiv.
- `<add>`: Zusätzlicher Code für das `<select ..>`-Tag

2.3.28 CONTACTNOW

ToDo!

2.3.29 CORPCOL

`{{CLTCOL<cltsuffix>}}` wird ebenso behandelt wie `{{ADD..}}`. Bezieht sich aber auf die Standardfarbe des Zielmandanten. Zurückgeliefert wird ein String der Länge 6. Dieser codiert die Farbe hexadezimal in der Form `RRGGBB`.

2.3.30 D

`{{D}}` wird ersetzt durch `<span class='dist' />`
Kann über CSS dazu benutzt werden, um einen horizontalen Textabstand zu erzwingen.

2.3.31 DATE

Wandelt einen Unix-Timestamp in ein Datum um und zeigt es an.

`{{DATE[<time>[;<shift>]]}}`

- `<time>`: Unix-Timestamp[1]. Default ist der Zeitpunkt zu dem die Seite angefordert wird.
- `<shift>`: Boolean. Falls `true` wird der angegebene erste Parameter `time` dem aktuellen Zeitwert aufaddiert und das Ergebnis als Datum dargestellt. Sonst nicht. Default ist 0.

`{{DATE}}` wird durch die Rückgabe von `date('d.m.Y',..)` ersetzt.

Siehe auch: TIME

2.3.32 DD

`{{DD}}` wird ersetzt durch `<span class='dist' /><span class='dist' />` Kann über CSS dazu benutzt werden, um einen horizontalen Textabstand zu erzwingen.

2.3.33 DWN

Alias für das Opening-Tag eines Download-Anchors.

`{{DWN<did>[;<targetClient>[;<noTag>]]}}`

- `<did>`: Download id des gewünschten Downloads innerhalb der Download-Tabelle des aktiven Mandanten.
- `<targetClient>`: Oprional. Client in dessen Download-Tabelle und unter dessen Download-Pfad die Zieldatei registriert ist. Per Default der aktive Client.
- `<noTag>`: Optional. Boolsch. Falls `true` wird nur die Url auf die Zieldatei zurückgeliefert. Sonst ein komplettes HTML-Anchor-Tag. Per Default `false`.

`{{DWN..}}` wird ersetzt durch einen mit `title` versehenen Anchor-Opening Tag der Form

`<a title='..' href='..'>`

Der Wert der `href` wird vorher durch eine regular Expression gefiltert.

[1] Die Unix-Zeit zählt die Sekunden seit dem 1. Januar 1970, 0.00 Uhr, UTC

2.3.34 DWN*

Variation von `{{DWN}}`. Diese Version des Aliases ist für PDF-Dokumente gedacht, die in einem neuen Fenster geöffnet werden sollen. Dafür kenne ich im Augenblick leider noch keine W3C-XHTML1.0/STRICT standardkompatible Möglichkeit.

2.3.35 EML

`{{EML<cltsuffix>}}` wird ebenso behandelt wie `{{ADD..}}`. Bezieht sich aber auf die E-Mail-Adresse des Zielmandanten.

2.3.36 FAX

`{{FAX<cltsuffix>}}` wird ebenso behandelt wie `{{ADD..}}`. Bezieht sich aber auf die FAX-Nummer des Zielmandanten.

2.3.37 FILE

Vereinfachte Version des Aliases `{{DWN..}}`. `{{FILE..}}` kommt ohne Download-Tabelle aus und ist für einmalig auftretende Gelegenheitsdownloads wie die Druckversion eines Artikels als statisches PDF-Dokument gedacht.

`{{FILE<fname>[;<cltSuffix>[;<title>]]}}`

- `<fname>`: Unterpfad und Dateiname der Zieldatei im Unix-Format, wobei direkt mit dem ersten Pfadnamen oder dem Dateinamen begonnen wird. Also `Unterpfad/text.pdf` anstelle von `./Unterpfad/text.pdf`. Außerdem wird davon ausgegangen, dass der Startpfad das Download-Stammverzeichnis des unter `cltSuffix` angegebenen Mandanten ist.
- `<cltsuffix>`: Kürzel des Mandanten in dessen Download-Pfad das Ziel liegt. Per Default ist der aktive Mandant eingestellt.
- `<title>`: Title für den Link. Per Default leer.

Das Alias wird ersetzt durch einen Opening Tag der Form

`<a class='file' href='http..' title='..'>`

2.3.38 FORM

Alias für das Opening-Tag einer Formularumgebung im eingestellten Standard-Charset (UTF-8). Die Defaulteinstellungen der Parameter erzeugen ein Formular für die Redaktion.

`{{FORM[<action>[;<method>[;<suffixes>[;<class>[;<add>]]]]]}}`

- `<action>`: `action`-Wert für das `<form>`-Tag. Default ist `{{PHPself}}`

- `<method>`: `method`-Wert für das `<form>`-Tag. Default ist `post`
- `<suffixes>`: String. Wird an die Action des Formulars angehängt. Falls 'this' übergeben wird wird der `$_GET`-Hash als GET-Parameter aufbereitet und angehängt. Default ist 'this'.
- `<class>`: HTML-Klassenbezeichner für die Formularumgebung. Per Default `FORM`.
- `<add>`: Zusätzliches HTML für die das `<form ..>`-Tag. Per Default leer.

Das Alias wird ersetzt durch

```
<form action='..' method='..' accept-charset='..'>
```

2.3.39 FORMe

`{{FORMe}}` wird ersetzt durch `</form>`

2.3.40 GOOGLEMAP

Alias für die Erzeugung eines Google Map-Fensters. Dieses wird in ein `<div>` der Klasse `GOOGLEMAP` eingebunden. Die Einbindung mehrerer solcher Fenster in einem Artikel ist möglich.
Die Karten werden auf Geokoordinaten im Gradmaß zentriert. Diese haben die Form (lat, lon) mit $lat \in [-90, 90]$ und $lon \in [-180, 180]$. Hierbei werden negative Breite für die Südhalbkugel der Erde und negative Länge für ihre westliche Hemisphäre verwendet.

```
{{GOOGLEMAP<(mixed)where>[;<(int)height>[;<(string)id>]]]}}
```

- `<where>`: Dieser Parameter ist eine ','-Liste und kann eine der beiden folgenden Formen haben:
  1. `<cltSuffix>[,<Punktnummer>]`: In den Kunden-Tabellen können mehrere Koordinatenpaare hinterlegt sein. Diese werden für den angegebenen Client ausgelesen und der angegebene Punkt wird verwendet. Die Zählung der Punkte beginnt bei null. `0` ist auch der Default für `Punktnummer`.
     Bsp.: `kh` oder auch `kh,0` beschreibt den Ort des Hauptsitzes von Koch Management Consulting.
  2. `<lat>,<lon>`: Ein Koordinatenpaar, wie eingangs beschrieben. Achten Sie darauf, dass die Zahlen amerikanisch notiert sind. Dezimalen werden also nicht, wie Deutschland üblich, mit einem Komma, sondern mit einem Punkt abgetrennt.
     Bsp.: `48.7419,9.1061` beschreibt einen Punkt in Stuttgart.

- `height`: Google-Zoom-Wert. Per Default 13. Auf dieser Auflösung kann man bereits Straßen und einzelne Gebäude gut erkennen. Höhere Werte zoomen noch näher heran (soweit bei Google Maps verfügbar), niederigere zoomen heraus. Ein Wert von 8 etwa gibt einen guten Überblick über die Region.

- `id`: Das `<div>` in welches die Karte eingebunden wird benötigt eine eindeutige HTML-id. Per Default wird diese fortlaufend als `GOOGLEMAP0`, `GOOGLEMAP1`,... nummeriert. Mit diesem Parameter können Sie die id selbst wählen.

2.3.41 GOTONEXT

Alias für einen Link auf den „nächsten“ Artikel in Gestalt eines gelben, abwärts gerichteten, mit einem statischen Inhalt der Form „Zum nachfolgenden Artikel“ beschrifteten Pfeils. Sowohl der Pfeil als auch der statische Inhalt ist von einem Div umgeben.

```
{{GOTOTOP[<(char)type>[;<(string)clt>[;(string)con]]]}}
```

- `<plain>`: Falls `b` (both sides) Es wird auf den vorangehenden und den nachfolgenden Artikel verlinkt.
  Falls `bf` (both sides first) Es wird nur auf den nachfolgenden verlinkt.
  Falls `bl` (both sides last) Es wird nur auf den den vorangehenden Artikel verlinkt.
  (both sides) ignoriert die Parameter `clt` und `con`.
  Falls `p` (text free) Gelber Pfeil nach unten ohne statischen Inhalt.
  Falls `n` (narrow) werden ein nach unten weisender Keil und das Wörtchen „weiter“ eingebunden. In diesem Fall erhält das Alias ein Inline-Div.
  Per Default hat der Pfeil die oben in der Kurzbeschreibung genannte Form.

- `<clt>`: Falls der Zielartikel nicht automatisch bestimmt werden soll, kann er auch explizit angegeben werden. `clt` steht hierbei für den Target Client. Default ist der aktive Client.

- `<con>`: Index Id des zu verlinkenden Artikels. Per Default der von `Kdb->getNextId(..)` zurückgelieferte Wert.

2.3.42 GOTOTOP

Alias für einen Link zum Anker `#top` in Gestalt eines gelben, aufwärts gerichteten, mit einem statischen Inhalt der Form „Zum Seitenanfang“ beschrifteten Pfeils.

```
{{GOTOTOP[<(char)plain>[;<(mixed)force>]]}}
```

- `<plain>`: Falls `p` wird nur ein gelber, aufwärts weisender Pfeil eingebunden, der zum Anker `#top` verlinkt.
- `<force>`: Falls `f` wird der Pfeil angezeigt. Falls eine Zahl wird der Pfeil nur angezeigt, falls diese Zahl > 15 ist. Die Stilklasse `standardDisp` übergibt an dieser Stelle ein Maß für die Artikellänge. Das führt dazu, dass der Pfeil nur dann angezeigt wird, falls der Artikel eine gewisse Mindestlänge überschreitet.

2.3.43 GTEXT

Schreibt Texte als png-Grafik mit transparentem Hintergrund.

`{{GTEXT<text>[;<color>[;<size>[;<width>[;<height>]]]]}}`

- `<text>`: Darzustellender Text. Damit auch das Semikolon als Zeichen möglich wird, wird das erste Zeichen des Textes als Delimiter verwendet. Der zu schreibende Text endet, sobald dieses Zeichen ein zweites mal auftaucht.
  Z.B. `'Hallo Welt!'` würde somit als `Hallo Welt!` interpretiert werden.
- `<color>`: Ein hexadezimaler RGB-Farbcode der Form `RRGGBB`. Per Default `000000`.
- `<size>`: PHP-Schrift-Font. Ein Wert aus `{1,..,5}`. Per Default `4`.
- `<width>`: Breite der zu erstellenden Grafik in Pixeln. Per Default 8 mal so viele wie der darzustellende Text Zeichen hat.
- `<height>`: Höhe der darzustellenden Grafik in Pixeln. Per Default 16.

2.3.44 HIDDEN

Alias für eine versteckte Inputbox. Wird von der Redaktion verwendet, um unter anderem den `action`-Parameter zu posten.

`{{HIDDEN<value>[;<name>]}}`

- `<value>`: `value` der Box.
- `<name>`: `name` der Box. Default ist 'action'.

Ersatz durch `<input type='hidden' name='..' value='..' />`

2.3.45 HOME

`{{HOME...}}` wird ersetzt durch eine Linkadresse auf eine Homepage des dem `cltsuffix` zugeordneten Kunden. Diese ist in der Client-Tabelle hinterlegt.

`{{HOME<cltsuffix>[;<id>]}}`

- `<cltsuffix>`: Client Suffix des Kunden dessen Url ausgegeben werden soll.
- `<id>`: Ganze Zahl. Beginnend bei 0 gibt sie die Nummer der Ziel-Url an. Das betreffende Feld in der Client-Tabelle ist eine `\r\n`-Liste in deren Zeilen verschiedene URLs stehen können. Dabei gilt im Karrierehandbuch die Konvention, dass die nullte Zeile auf die externe Homepage des Unternehmens und die erste ggf. auf die firmeneigene Stellenbrse verweist. Default ist 0.

2.3.46 IMG

Alias für die Einbindung einer Mandantengrafik. Diese Grafik muss im Images-Unterverzeichnis des Mandanten liegen.

`{{IMG<fname>[;<cltsuffix>[;[div][;<url>]]]}}`

- `<fname>`: File Name der Grafik. Kann aber auch ein String der Form `subpath/fname` sein. Der Parameter wird durch eine Regular Expression gefiltert. Insbesondere Zeichenfolgen '..' sind nicht zugelassen.
- `<cltsuffix>`: Die Grafik wird im Verzeichnis `../Images/<cltSuffix>` gesucht. Per Default wird das Clientsuffix des aktiven Mandanten verwendet. Eine Ausnahme entsteht, wenn ein bestimmter Darstellungs-GET-Parameter `media` (z.B. `media=handheld`) verwendet wird. In diesem Fall prüft das Alias, ob die gesuchte Datei auch in einem Verzeichnis `../Images/<cltSuffix>/<media>` vorhanden vorhanden ist und lädt bevorzugt diese.
- `div`: Wird als dritter Parameter die case insensitive Zeichenfolge `div` übergeben wird die Ausgabe zusätzlich in `<div class='IMG'>..</div>` eingebunden.
- `<url>`: Falls angegeben wird die Grafik gleichzeitig ein Link auf diese URL.

Falls alle Optionen wahrgenommen werden wird das Alias ersetzt durch

```
<div class='IMG'><a class='IMG' href='..'>
  <img class='IMG' alt='..' src='..' /></a></div>
```

Als `alt` wird der übergebene Dateiname verwendet.

2.3.47 JOB

Erstellt einen Link zu einer aus einer `job_<lng>_<clt>`-Tabelle heraus automatisch erzeugten Stellenausschreibung.

`{{JOB[<clt>[;<id>[;<lng>[;<type>[;<title>]]]]]}}`

- `<clt>`: Client-Suffix des Mandanten, dem die auszuwertende `job`-Tabelle zugeordnet ist. Per Default der aktive Client.
- `<id>`: Primary id der auszuwertenden Stellenbeschreibung. Per Default die id der ersten Zeile in der Tabelle.
- `<lng>`: Sprachsuffix der Sprache die der auszuwertenden Tabelle zugeordnet ist. Per Default die aktive Sprache.
- `<type>`: Art des Links. Bislang implementiert:
  - `link`: Kompletter Link mit Opening-Tag, Closing-Tag und Linktext (Default).
  - `opening`: Nur der Opening-Tag für den Link.
  - `plain`: Nur der Title. Kein Link wird gesetzt.
- `<title>`: Text für den Link. Per Default der Title zu verlinkenden Jobs.

2.3.48 JOBLIST

Erstellt `{{JOB..}}`-Links für alle aktiven Stellenangebote einer Sprache eines Mandanten.

`{{JOBLIST[<clt>[;<lng>[;<type>[;<sep>]]]]}}`

- `<clt>`: Client-Suffix des Mandanten, dem die auszuwertende `job`-Tabelle zugeordnet ist. Per Default der aktive Client.
- `<lng>`: Sprachsuffix der Sprache die der auszuwertenden Tabelle zugeordnet ist. Per Default die aktive Sprache.
- `<type>`: Type für die darzustellenden `{{JOB..}}`-Links. Wird direkt in die erzeugten `{{JOB..}}`-Aliase eingefügt. Per Default leer.
- `<sep>`: Separator der zwischen die einzelnen `{{JOB..}}`-Aliase eingefügt wird.
  Achtung: Die Syntax verlangt eine Begrenzung durch Hochkommas: `'<sep>'` und erlaubt somit das Verwenden von Semikoli `;` im Separator-Code. Default ist `\r\n` für einen Zeilenumbruch um Carriage Return.

Beispiel: `{{JOBLISTkoch;;;' {{WRY}} '}}` erzeugt eine durch gelbe Keile getrennte Liste der Jobs des Mandanten `koch` aus Daten der der aktiven Sprache zugeordneten Job-Tabelle.

2.3.49 JUS

`{{JUS<cltsuffix>}}` wird ebenso behandelt wie `{{ADD..}}`. Bezieht sich aber auf die juristische Bezeichnung des Zielmandanten.

2.3.50 KBOOK

Analog zu `{{BBOOK}}`, wird allerdings durch ein grünes Buch-Thmub-Nail ersetzt, welches auf den Artikel `con=29010201,clt=kh` (die Download-Seite des Karrierehandbuchs) verlinkt.

2.3.51 KH_URL

`{{KH_URL}}` wird ersetzt durch

`{{PHPi_protocol}}{{PHP_SERVER;HTTP_HOST}}/{{PHPi_main_page}}`

2.3.52 MPFORM

Multipart-Formular-Umgebung. Im Grunde das Alias `{{FORM..}}` mit der zusätzlichen Option `enctype='multipart/form-data'`.

Wird für File-Upload-Formulare benötigt.

2.3.53 NAME

`{{NAME<cltsuffix>}}` wird ebenso behandelt wie `{{ADD..}}`. Bezieht sich aber auf den Namen des Zielmandanten.

2.3.54 OBJECT

`{{OBJECT..}}` erzeugt eine Umgebung in der Applets wie z.B. ein Youtube-Video eingebunden werden können. Bedient sich sowohl HTML-`<object..>` als auch `<param..>`-Tags.

`{{OBJECT[<nogo>[;<add>[;<param>]]]}}`

- `nogo`: Code der hinter die optionalen Parameter im Object-Tag abgelegt werden soll.
- `add`: Code der Innerhalb des Attributbereichs des Object-Tags abgelegt werden soll.
- `param`: `','`-Liste von Schlüssel-Werte-Paaren der Syntax `key=>value`. Diese Paare werden HTML-in `param`-Tags umgesetzt mit dem Name `key` und dem Value `value`.

Beispiele:

```
{{OBJECTFlash-Video von Harald Lesch;codetype="application/
  x-shockwave-flash" height="344" width="425";
  movie=>http://www.youtube.com/v/MbvMVrkhaRo&amp;hl=de&amp;fs=1,
  allowFullScreen=>true,allowscriptaccess=>always}}
```

wird zu

```
<object codetype="application/x-shockwave-flash" height="344"
  width="425"><param name='movie'
  value='http://www.youtube.com/v/MbvMVrkhaRo&amp' />
  Flash-Video von Harald Lesch</object>
```

Oder

```
{{OBJECTJava-Rubik-Cube von Karl Hoernell;
  codetype="application/java" classid="java:rubik.class"
  standby="One moment, Sir." codebase='Extern/Rubik'}}
```

Wird zu

```
<object codetype="application/java" classid="java:rubik.class"
  standby="One moment, Sir." codebase='Extern/Rubik'>
  Java-Rubik-Cube von Karl Hoernell</object>
```

Die Zeilenumbrüche sind hierbei nachträglich eingefügt worden.

2.3.55 PAR

`{{PAR}}` wird ersetzt durch das Opening-Tag eines Anchors auf die `url` des ersten Parent-Artikel des aktiven Artikels. Außerdem wird die `id` des aktiven Artikels als Sub-Anchor verwendet.

```
<a href='url#id'>
```

2.3.56 PDF

Hat im HTML-Modus keine Auswirkung. Sein Parameter ist aber ein Steuerbefehl für den PDF-Konverter.

```
{{PDF<command>}}
```

Bislang unterstützte Kommandos:

- `newpage`: Erzwingt einen Seitenumbruch.

Hinweis: Das PDF-Alias bildet einen eigenen Block. Dies kann zu unerwünschten Effekten führen, falls dadurch kohärente Textblöcke zerteilt werden. Fügen Sie das Alias nach Möglichkeit nur zu Zeilenbeginn oder -ende ein.

2.3.57 PHP

Wird ersetzt durch den Wert einer als Parameter übergebenen Variablen.

`{{PHP<varName>[;<key>]}}`

- `<varName>`: Key eines Wertes der entweder im Hash `$GLOBALS` oder in einem `htDisplay->replaceAliases(..)` als `$extra` übergebenen Hash zu finden ist.
- `<key>`: Optionaler Zusatzparameter. Falls angegeben geht das Programm davon aus, dass die Zielvariable selbst ein Hash ist und gibt denjenigen Wert zurück der unter `<varName>[<key>]` gefunden wird.

Falls der gesuchte Wert nicht gefunden werden kann wird das Alias einfach entfernt.

Dieses Alias stellt natürlich ein gewisses Sicherheitsrisiko dar. Aus diesem Grund werden einige als *besonders vertraulich* eingestufte Variablen vor der Auswertung gesondert herausgefiltert und sind über `{{PHP..}}` nicht darstellbar.

2.3.58 REM

`{{REM<irgend_ein_Kommentartext>}}` wird als 'remark' ersatzlos entfernt.

2.3.59 SA

Das Alias wird ersetzt durch das HTML einer Short Advertisement Visitenkarte.

`{{SA[<clt>[;<saId>]]}}`

- `<clt>`: Client Suffix des Kunden dem die darzustellende SA zugeordnet ist. Per Default ist der aktive Kunde eingestellt. Wird hier `RAND` angegeben, wird ein Zufallskunde aus den Standard-Clients ausgewählt.
- `<saId>`: Id der darzustellenden Visitenkarte innerhalb der SA-Tabelle des Zielkunden.

2.3.60 SATAR

Dieses Alias liefert eine Liste von Links auf Ziele von Visitenkarten zurück.

`{{SATAR[clt[;sid[;demo]]]}}`

- `<clt>`: Suffix des Mandanten für den die Liste erstellt werden soll. Default ist `kh`.

- `<sid>`: Id der Visitenkarte innerhalb der Short-Ad-Tabelle des Mandanten. Falls nicht angegeben oder mit `all` belegt werden alle Visitenkarten gleichermaßen berücksichtigt. Default ist `all`.
- `<demo>`: Optional. Kann einen in Apostrophe `'...'` geklammerten `cid`-String im Stil der `sa_<lng>_<clt>` Tabellen enthalten. In diesem Fall werden die tatsächlichen Target Ids ignoriert und stattdessen die Demo-Liste verwendet. Dies ist sinnvoll für die Demonstration einer Probe-Site bei einem potentiellen Neukunden. Per Default ist dieser Feature deaktiviert.

Zurückgeliefert wird ein zeilenweise organisierter String der Form

```
{{CLT<clt>;<cid>;;0}}<tclt==kh?'Kapiteltitel - ':''><title(cid)>{{ae}}
...
```

2.3.61 STATIC

Dieses Alias erlaubt die Verwendung von Inhalten aus den Static-Tabellen.

`{{STATIC<handle>[;<context>[;<args>]]}}`

Die Parameter werden direkt an `staticTools::fmtStatCon(..)` weitergeleitet. Insbesondere darf der dritte Parameter, `args`, Semikoli enthalten.

2.3.62 TEL

`{{TEL<cltsuffix>}}` wird ebenso behandelt wie `{{ADD..}}`. Bezieht sich aber auf die Telefonnummer des Zielmandanten.

2.3.63 TIME

`{{TIME<timestamp>[;<shift>]}}` wird ebenso behandelt wie `{{DATE..}}`. Bezieht sich aber auf die Uhrzeit, die in der Form `HH:MM:SS` dargestellt wird.

2.3.64 TITLE

`{{TITLE}}` wird durch den `title`-Inhalt eines Contents ersetzt.

`{{TITLE[<tclt>[;<id>]]}}`

- `<tclt>`: Target Client. Default ist der aktive Client.
- `<id>`: Index-Id des Zielartikels. Default ist der aktive Artikel.

2.3.65 TITLELINE

Fügt einen Überschriftenlink auf den ersten Parent-Knoten des aktiven Contents ein.

`{{TITLELINE[<title>]}}`

- `<title>`: Text `title` für den Überschriftenlink. Default ist '{{TITLE}}'

Das Alias wird ersetzt durch

```
<div class='titleline'><a href='..'>
  <span class='i'>|</span>  
    title
    <span class='i'>|</span>
</a></div>
```

2.3.66 TITLELINE*

Variation von `{{TITLELINE}}`. Verhält sich identisch zum Original, legt aber keinen Link an.

2.3.67 TOC

Inline Table of Contents. Erzeugt einen als `CLmi` formatierten Table of Contents der Tiefe 2. Bezieht sich auch auf den aktiven Client.

`{{TOC[<id>[;<includeRoot>[;<doClFrame>[;<firstNoLink>]]]]}}`

- `<id>`: Index-Id des Artikels innerhalb der Indextabelle des aktiven Clients. Per Default die aktive Id.
- `<includeRoot>`: Boolsch. Soll die Wurzel bereits Teil des TOC sein. Per Default `false`
- `<doClFrame>`: Boolsch. Soll eine Checklist-Umgebung um die Ausgabe gesetzt werden? Dies ist erforderlich, falls das Alias nicht bereits in einer Checkliste platziert ist. Default ist `false`.
- `<firstNoLink>`: Boolsch. Soll die erste Ebene des Verzeichnisses nur Plaintext ohne Link sein? Per Default `false`.

Beispiele:

1. Die Code-Fragmente

   ```
   CLmi
   {{TOC}}
   CLe
   ```

und

```
{{TOC;;true}}
```

erzeugen einen Standard-ToC, aufbauend auf dem aktiven Artikel.

2. Die Code-Fragmente

```
CLmi
{{TOC100;true}}
CLe
```

und

```
{{TOC100;true;true}}
```

erzeugen jeweils ein Inhaltsverzeichnis mit dem Artikel 100 und seinen eventuell vorhandenen Kids als einzigen Ast.

Anmerkung an den Admin: Damit dieses Alias richtig arbeitet, muss es vor dem Aufruf von `htDisplay->std2html()` *ausgewertet werden.* `htDisplay->importContents()` *sorgt dafür. Das soll so bleiben!*

2.3.68 TR

Tabellenzeilenumgebungseröffnungstag.

```
{{TR[<class>]}}
```

- `<class>`: CSS-Klasse der Zeile. Per Default ein bei 'even' beginnender Wechsel zwischen 'even' und 'odd'. Die Buchführung funktioniert über eine statische Variable. Es wird auch umgeschalten falls ein Wert für `<class>` angegeben wurde.

Das Alias wird ersetzt durch `<tr class='..'>`

2.3.69 TRe

`{{TRe}}` wird ersetzt durch `</tr>`

2.3.70 USER

Wird durch öffentliche Informationen über einen bestimmten User ersetzt.

```
{{USER<login>[;<datakeys>]}}
```

- `<login>`: Login des Users.

- `<datakeys>`: Eine ;-Liste anzuzeigender Daten. Zugelassene Schlüssel sind
  - `address`: Anrede. Meist „Herr“ oder „Frau“.
  - `firstname`: Vorname.
  - `lastname`: Nachname.
  - `tel`: Gegebenenfalls die *öffentliche* Telefonnummer des Users.
  - `appmail`: Gegebenenfalls die *öffentliche* E-Mail-Adresse des Users.
  - `times`: Gegebenenfalls Sprechzeiten.
  - `url`: Die in der Kontakttabelle veröffentlichte Url.
  - `urlcaption`: Der zu obiger Url gehörige Link-Title.
  - `onlinetext`: Der Online-Text des Users.
  - `offlinetext`: Der Offline-Text des Users.

  Werden mehrere Schlüssel übergeben, wird das Alias durch eine durch Spaces getrennte Liste der geforderten Daten ersetzt.

Beispiel:
`{{USERkochmn;firstname;lastname}}` wird zu „Markus-Hermann Koch“.

2.3.71 WRY

'Wedge Right Yellow'. `{{WRY}}` wird ersetzt durch

```
<span class='gelb'>&#9658;</span>
```

`&#9658` erzeugt eine ausgefüllte Version des Symbols ▷.

2.3.72 WDY, WLY, WUY

Analog zu 'Wedge Right Yellow': 'Wedge Down Yellow', 'Wedge Left Yellow' und 'Wedge Up Yellow'.

3 Das Back End KARRIEREredaktion

3.1 Motivation

Mit der Einführung des Mandantensystems kam nicht nur für die einzelnen Kunden die Möglichkeit ins Spiel eigene Sub-Sites zu besitzen, sondern es wurde auch erforderlich diesen Kunden eine Möglichkeit an die Hand zu geben ihre Inhalte weitgehend selbst zu verwalten.

Mit anderen Worten: Ein einigermaßen mächtiges, ansprechendes und gegen Angriffe von außen gesichertes Back End musste her.

3.2 Secure Sockets Layer

Derzeit wird die Site von 1&1 mit dem Tarif Hompage Business gehostet. In diesem Angebot enthalten ist die kostenlose Ausstellung eines von Equifax unterzeichneten SSL-Zertifikates.

Da gerade für die Belange des Kundendaten enthaltenden Backends eine gesicherte Übertragung der Daten gegeben sein sollte, ist die Redaktion das Angebot annehmend auf `https://` gesetzt worden.

Dafür erforderliche Einstellungen in den PHP-Skripten:

1. In den Dateien `~/HtTools/redaktion.php5` und `~/HtTools/fgMain.php5` wurde die in `~/kinit.php5` definierte Variable `$i_protocol` neu belegt: `$i_protocol = 'https://';`

2. In `~/kinit.php5` selbst wurde die Variable `$i_logon_cookie_secure` von `NULL` auf `true` gesetzt. Dies bewirkt, dass das von der Redaktion erzeugte Cookie so eingestellt wird, dass der Browser es nur dann zurücküberträgt, wenn die Verbindung gesichert ist. Die Variable wird bislang nur in `~/Scripts/authentifier.php5` gebraucht.

3.3 Das Rechtesystem

Rechte eines Benutzers sind immer an Mandanten gebunden. Vergleiche hierzu im Abschnitt 1.3.12 zur Struktur der Tabelle `karriere.authority` die Merkmale `cltsuffix` und `auth`.

Abbildung 3 zeigt die Hierarchie des Rechtesystems der Redaktion.

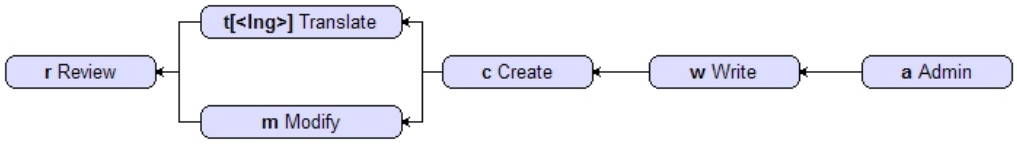

Abbildung 3: Die Hierarchie der Rechte

Es folgt eine Beschreibung der einzelnen Rechte. Höhere Rechte umschließen dabei sämtliche Funktionen aller ihnen untergeordneten Rechte.

- `r`: *Review* ist das kleinste Recht und dazu gedacht, statistische Werte überprüfen zu können. User mit diesem Recht können Datenbankkonsistenztests durchführen. Außerdem sind sie vollwertige User, die zum Beispiel Sprechzeiten haben können.

- `m`: *Modify* erlaubt die Modifikation bestehender Inhalte des rechtegewährenden Mandanten. Die Baumstruktur darf dabei nicht angefasst werden. 'Modifikation' erstreckt sich auch nicht auf die vollständige Entfernung von Inhalten.

- `t[<lng>]`: *Translate*. Dem Übersetzerrecht ist ein Sprachkürzel nach ISO 639-1 zugeordnet. Ein Übersetzer kann bestehende Inhalte der betreffenden Sprache ändern. Außerdem kann er über die Übersetzermaske neue Inhalte anlegen sofern bereits ein Originaleintrag in einer anderen Sprache existiert der übersetzt werden kann.

- `c`: *Create*. Wer das kreative Recht besitzt kann auf den betreffenden Clients neue Knoten in der Baumstruktur anlegen.

- `w`: *Write*. Ein Benutzer mit Schreibrechten hat vollständige Rechte zum Lesen, Schreiben und Löschen von Inhalten, Knoten und Visitenkarten innerhalb der Karriere-Sub-Site der betreffenden Clients.

- `a`: *Admin*. Admins haben vollständige Rechte auf den betreffenden Clients. Sie können Benutzer anlegen, ändern und entfernen sofern sich die Rechte dieser Benutzer in Clients erschöpfen die von dem betreffenden Admin administriert werden.

3.4 Die Gültigkeit von Cookies

Nur wer ein als gültig anerkanntes Cookie besitzt kann mit den Rechten des dem Cookie zugeordneten Benutzers auf der Redaktion Aktionen ausführen.

Der Value des Cookies ist eine `;`-Liste. Sie setzt sich zusammen aus

- Dem Login des Users.

- Dem Unix-Timestamp des Zeitpunktes an dem das Cookie seine Gültigkeit verlieren soll.
- Einer 32 Zeichen langen Zufallsfolge von Alpha-Numericals.

Über die Gültigkeit des Cookies wacht die Methode `authentifier->isValidCookie()`. Sie prüft ab, ob

1. Überhaupt ein Cookie vorliegt.
2. Das Cookie sich mit dem Login eines Users identifiziert.
3. Dieser User zur Zeit in der Tabelle `karriere.online` eingetragen ist.
4. Dieser Eintrag noch nicht älter als die Lebensdauer eines Cookies ist. Diese wird in der globalen Variablen `i_logon_cookie_duration` in Sekunden festgelegt und beträgt im Augenblick zwei Stunden. Redundant wird auch die vom Cookie angegebene Lebensdauer überprüft.
5. Ebenso wird die Zufallssignatur des Cookie-Wertes mit der in der Online-Tabelle abgelegten Version verglichen.
6. Schließlich wird überprüft, ob die ip von der aus eine Anfrage an die Redaktion gestellt wurde (und bereits die Öffnung eines Formulars zur Änderung von Inhalten zählt als Anfrage) mit der ip übereinstimmt von der aus das Cookie ursprünglich beantragt wurde.

Wenn einer der Punkte verletzt ist wird das Cookie nicht akzeptiert.

Die von der Redaktion an den Browser übersandten Cookies werden lediglich im Speicher des Clients abgelegt. Sie verschwinden bei Beendigung des Browsers.

3.5 SQL-Injections

Eine beständige Gefahr der sich externen Anwendern ausgesetzte Datenbanken gegenübersehen sind Angriffe durch SQL-Injection. Für die Neufassung des Karrierehandbuches schafft hier die statische Methode `staticTools::hackSecureEncode()` Abhilfe.

Sämtliche aus welchen Gründen auch immer dubiosen Einträge können mit dieser Methode vor ihrem Einbau in SQL-Queries entschärft werden.

Das funktioniert so: Es existiert eine White List „harmloser“ Zeichen. Dies sind vornehmlich die Alpha-Numericals sowie einige weitere UTF-8-Words – vornehmlich die deutschen Sonderzeichen und bislang die kyrillische Schrift.

- Jedes auf der Liste enthaltene Zeichen wird unverschlüsselt weitergeleitet.

- Für alle anderen Zeichen werden die Ordnungszahlen der beteiligten Bytes bestimmt und deren Characters durch das Alias `{{CHR..}}` ersetzt.

Unter anderem nicht *white listed* sind die diversen Akzente, Anführungszeichen, das für SQL-Kommentare benötigte Minuszeichen, Zeilenumbrüche und das Semikolon.

4 Konventionen

4.1 Metakonventionen

Bei der Neufassung des Karrierehandbuchs wurden von Beginn der Planung an folgende Ziele verfolgt:

a) **Redundanz vermeiden.**

b) Umstrukturierung und Wartung der Inhalte sollen problemlos möglich sein.

c) Jedem Inhalt soll eine eindeutige URL zugewiesen sein.

d) Dabei soll auch Mehrsprachlichkeit unterstützt werden.

e) Die Formatierung der Inhalte soll flexibel sein. Portabilität auf Mobilfunktelefone ist ein Fernziel.

f) Die Site ist suchmaschinenfreundlich. Insbesondere die Frame-Technologie der ersten Version soll weichen.

g) Der Code soll auch nach längerem Liegenlassen pflegeleicht sein.

h) Die Site soll ggf. einen Serverwechsel ohne großen Aufwand überstehen.

i) Die Site soll auf einigermaßen modernen Browsern gut aussehen und auf älteren Browsern zumindest gut lesbar sein.

j) All user input is evil, until proven otherwise.

Diese Liste im Blick haben sich im Laufe der Zeit folgende Richtlinien ergeben.

1. W3C-Standard ist XHTML 1.0 Strict.

2. Trennung von Inhalt und Quellcode.
   Textinhalte gehören in die Datenbank.

3. Innerhalb der Datenbank: Trennung von Inhalt und Struktur. Gerade Übersetzungen verwenden dieselbe Struktur wie das Original und sollten diese nicht redundant kopieren.

4. Trennung von Inhalt und Formatierung. Formatierungen erfolgen so weit es geht über ausgelagerte Cascading Style Sheets.

5. Trennung von thematisch unzusammenhängenden Inhalten.

4.2 Konkrete Regeln

4.2.1 `kinit.php5` und Umgebung

- `kinit.php5` wird immer zuerst eingebunden. `kinit.php5` bindet das plattformabhängige Initialisierungsskript `init<Plattform>.php5` ein. Dies geschieht mit Hilfe der Variablen `$chooseInit`
- Plattformunabhängige Konstanten die für mehr als eine Datei Belang haben werden in `kinit.php5` als globale Variablen definiert. Sie erhalten Namen die mit den Zeichen `i_` beginnen.
- GET-Parameter enthalten nur Alpha-Numericals und werden bereits von `kinit.php5` entsprechend gefiltert. Diese Vorgehensweise ist nicht praktisch für POST-Parameter. Letztere werden mit der entsprechenden Vorsicht weiterbehandelt.
- Eine Sonderrolle nehmen die Variablen `i_sub`, `i_con`, `i_lng` und `i_clt` ein. Sie werden aus den GET-Parametern heraus beschrieben.
- Globale Variablen werden von den Konstruktoren der großen Klassen ausgewertet. In den untergeordneten Methoden werden lokale Variablen verwendet. Statische Methoden bilden naturgemäß die Ausnahme.
- Mit der Boolschen Variablen `i_Debug` wird festgelegt, ob oder ob nicht das Programm im Debug-Modus läuft. Im Debug-Modus führen viele Unsauberkeiten die ansonsten abgefangen werden zu kontrollierten Abstürzen mit einigermaßen ausführlichen Fehlermeldungen.

4.2.2 Programmierung

- Jede Klasse erhält eine eigene `.php5`-Datei.
- In den ersten Zeilen dieser Datei werden via `require_once` alle Dateien eingebunden, die von der Klasse benötigt werden. Dieses Vorgehen ist bisweilen redundant erhöht aber die Übersichtlichkeit bei späterer Wartung.
- Vor der Klassendefinition steht eine Kurzbeschreibung von Funktionsweise und Zweck der Klasse.
- Unmittelbar nach dem Kopf werden sämtliche klassenglobale Variablen deklariert.
- Die erste Methode ist `__construct`. Sie initialisiert alle klassenglobalen Variablen. Sie ist auch (neben statischen Methoden) die einzige Methode die auf das Array `$GLOBALS` zugreift.

- Wenn nicht anders erforderlich werden Methoden als `private` oder `protected` deklariert. Funktionen von denen bekannt ist, dass sie nur lokal gebraucht werden sind leichter wartbar.

- Variablen der Klasse sind nicht direkt von außen zugänglich. Dazu sollen ggf. Trivialfunktionen `get<varName>()` oder `set<varName>(..)` dienen.

- Jede Klasse enthält die Variablen `debug` und `error`. Im Quellcode der Methode `Klasse->funktion(..)` wird mit abfangbaren Fehlern wie folgt oder auf vergleichbare Weise umgegangen:

  ```
  $this->error = "Klasse->funktion(): Was ging schief? ".
    "Variablenwerte!";
  if ($this->debug) die($this->error);
  ```

- Jede nicht vollkommen triviale Methode erhält nach der Definitionszeile

  ```
  function f(paramName_1,...,paramName_n)
  ```

  einen sie beschreibenden Kommentarblock folgender Gestalt:

  ```
  Ein- oder zweizeilige Zweckbeschreibung.
    (typ)paramName_1: Beschreibung.
    ...
    (typ)paramName_n: Beschreibung.
  ggf. Beschreibung des Rueckgabewertes.
  ```

  Bei der Bezeichnung der Parametertypen wird so getan, als wäre das Programm in einer stark typisierenden Sprache wie `C++` geschrieben.

- Kommentierung soll hilfreich sein. Es wird nicht gespammt, aber auch nicht mit Hinweisen gegeizt. Die Kommentierung soll als Dokumentation dienen können.

- Auch Variablenbezeichner sollten sprechend gewählt werden. Variablen mit einbuchstabigen Namen kommen vor, haben aber sehr kleine Geltungsbereiche.

- Quellcode wird durch 70 Zeichen lange Kommentarzeilen der Form

  ```
  //> Kurzbeschreibung Ebene 1. ------------------------
  Quellcode
  //>> Kurzbeschreibung Ebene 2. -----------------------
  Quellcode
  //<< -------------------------------------------------
  //< --------------------------------------------------
  ```

untergliedert. Alternativen sind denkbar.

- Bei der Operatorpräzedenz gilt die Punkt-vor-Strich-Regel der Grundrechenarten. Für alle anderen Fälle sollte mit Klammern gearbeitet werden.
- Der ternäre Operator

  ```
  <Bedingung> ? <true-Value> : <false-Value>
  ```

  wird nicht verschachtelt. Außerdem wird er nur verwendet, wenn auch sein `else`-Part einen nicht trivialen Zweck erfüllt.
- Nur in Ausnahmefällen enthalten Quellcodezeilen mehr als eine Anweisung.
- Quellcodezeilen haben höchstens 80 Zeichen.

4.2.3 Datenbank

- Jeder neue Tabellentyp wird in den dafür vorgesehenen `i_sql_`-Blocks in `kinit.php5` registriert.
- Jede Tabelle enthält eine Timestamp-Spalte für den Zeitpunkt der letzten Änderung und eine Spalte mit Textfeld `Comment` für Adminkommentare.
- Tabellen gleicher Struktur haben denselben Basisnamen. Sie werden durch Suffixe voneinander unterschieden. Die Reihenfolge innerhalb der Namensgebung lautet

  ```
  <Stammname>_<lng>_<clt>_...
  ```

- Noch nicht konsequent umgesetzt: Die Zeichen-Codierung ist UTF-8.

4.2.4 Pfade und Dateien

- Verzeichnisnamen beginnen mit einem Groß-, Dateinamen mit einem Kleinbuchstaben.
- Dateien und Verzeichnisse die im Namen ein Datum tragen sollen tragen es in der Form YYMMDD. Auf diese Weise werden die Dateien in der Regel nach Datum sortiert angezeigt.
- Im `htdocs`-Stammverzeichnis `~` liegen
  - `kinit.php5`
  - `init<Plattform>.php5`

  - `index.*`
  - `karrierehandbuch.php5`

- `~/Doc` enthält die Dokumentation. Diese Programmiererdokumentation liegt im Unterverzeichnis `MainDoc`, der Werbeflyer für die Neufassung liegt in `Flyer`.
- `~/Downloads` enthält für jeden Mandanten der Downloads bereit stellen will ein Unterverzeichnis welches als Namen den Clientsuffix des Mandanten verwendet. Diese Stelle ist eine Ausnahme von der Regel die besagt, Verzeichnisnamen stets mit einem Großbuchstaben beginnen.
- `~/Images` ist genauso strukturiert wie `~/Downloads`, enthält aber Zusätzlich noch das Unterverzeichnis `Thumbs` für die im Karrierehandbuch verwendeten Kleingrafiken.
- Alle Vorlagen außer `karrierehandbuch.php5` liegen in `~/HtTools`
- Alle Skripte und Klassen liegen in `~/Scripts`
- CSS liegen in `~/Styles`. Dort sind sie auf die Unterverzeichnisse `KarriereCSS` für das Front End und `HttoolsCSS` für die Back End Seiten verteilt. Eine Aufteilung die bislang eher Probleme verursacht als Vorteile eingebracht hat.
- `~/Tmp` ist ein Kurzzeittempverzeichnis dessen Inhalt jederzeits gelöscht werden kann.
- Im Verzeichnis `~/Backup` liegen selbige in Unterverzeichnissen `B<YYMMDD>[K]`. Das 'K' am Ende des Namens kommt hinzu, wenn es sich um ein Komplettbackup handelt. Ansonsten werden nur Dateien aufgenommen, die sich geändert haben. Auch im Komplettbackup fehlen externe Pakete wie `fpdf`. Diese haben ein eigenes Backup - Unterverzeichnis.

4.2.5 Abschließender Hinweis

- Achte die Regeln aber sei nicht blind.

5 Weitere Fragen – Kontaktadresse

Bei konsequenter wie in Abschnitt 4.2.2 beschriebener Kommentierung sollte der Quellcode keine weitere Dokumentation benötigen. Daher verweise ich Sie an dieser Stelle auf das eigentliche Programm.

Sollten Sie dennoch Fragen haben, fühlen Sie sich eingeladen mich persönlich zu kontaktieren. Am einfachsten per E-Mail unter

`mhk@karrierehandbuch.de`