1 Heinrich IV. und XML

von Bernhard Assmann, Köln

1.1 Einleitung

Wer sich mit mittelalterlichen Urkunden beschäftigt, dem wird schon im Proseminar beigebracht, daß es dafür nur eine ernstzunehmende Edition gibt, und zwar die der Monumenta Germaniae Historica. In diesem Vortrag geht es um Möglichkeiten, die Monumenta-Bände im Regal stehen zu lassen und statt ihrer und aus ihnen eine Digitale Edition zu erstellen. Die Vorteile dieser Editionsform sind im Rahmen dieser Tagung schon vielfältig diskutiert worden, so daß hier nicht mehr darauf eingegangen werden muß. Im Folgenden möchte ich zunächst ein kleineres Pilotprojekt vorstellen, das mittelalterlichen Königsurkunden gewidmet ist, und im Anschluß daran kurz auf XML und XSL(T) eingehen.

1.2 Die Speyrer Urkunden Heinrichs IV. im WWW

Für das Projekt wurden die Diplome Kaiser Heinrichs IV. für Speyer ausgewählt. Es sind 32 Diplome, die seine gesamte Regierungstätigkeit umspannen. Auch befinden sich verunechtete und gefälschte Diplome darunter. Damit stellt die Auswahl eine ideale Testumgebung für das Finden und Erproben neuer Möglichkeiten bei der Digitalen Edition von Diplomen dar. Als Textgrundlage diente der entsprechende Diplomata Band der MGH. Bei 20 der Diplome, nämlich jenen, die original überliefert sind, konnte eine Abbildung mit in das Projekt eingearbeitet werden. Die Originale liegen allesamt im Generallandesarchiv Karlsruhe und die Veröffentlichung der Bilder geschieht mit dessen Genehmigung. Die fotografischen Abbildungen wurden mir freundlicherweise von den Monumenten zur Verfügung gestellt.

1.3 Allgemeine Vorteile der Textauszeichnung

Um die Texte der Diplome digital zu speichern und zu bearbeiten, wurde die Methode der Textauszeichnung angewandt. Die Vorteile dieser Methode seien noch einmal kurz zusammengefaßt: Zu nennen sind neben Zukunftssicherheit vor allem Plattform- und Anwendungsunabhängingkeit und die Option auf vielfältige Nutzungsformen, da der Text als ASCII bzw. Unicode gespeichert wird.

Da die Auszeichnung mehr ist als die bloße Übertragung der Buchstaben des Textes in ein digitales Format, ist es außerdem möglich, zusätzliches Wissen, das mit dem Text verknüpft ist, mit zu speichern. Denn jede Auszeichnung, falls sie Sinn macht, ist dadurch charakterisiert, daß der Text innerhalb der Auszeichnungen näher beschrieben wird. So kann der Editor jedes strukturelle Element des Diploms auch explizit benennen, während im Buchdruck dieses Wissen bislang stets verloren ging. Gerade Urkunden bieten sich durch ihre angenommene innere Struktur für die Textauszeichnung an. Ein ausgezeichnetes Diplom, wie es im Projekt Verwendung gefunden hat, findet sich in Anhang 1.

Zu den im Dokument vorkommenden Auszeichnungen existiert eine Dokument Typ Definition (DTD), also eine Datei, in der die Grammatik der Auszeichnungen festgelegt wird. Sie ist bei der Arbeit gewissermaßen als Nebenprodukt angefallen. Zwingend

notwendig ist sie nicht. Es hat aber Vorteile, eine DTD zu formulieren, damit den Bearbeitern die Regeln klar vor Augen gestellt werden, nach denen die Dokumente auszuzeichnen sind. Geschieht dies nicht, steigt die Wahrscheinlichkeit, daß Fehler nicht erkannt werden. Diese können spätestens bei der Generierung der Ausgabe zu Folgefehlern führen, die nur sehr schwer zu finden sind. Wer aber die Absicht hat, sich mit anderen über seine Ergebnisse auszutauschen, kommt gar nicht umhin, eine DTD zu erstellen. Mit *anderen* sind dabei nicht nur Personen gemeint, sondern u.U. auch Computerprogramme. Denn erst wenn die Regeln bekannt sind, kann über die Auszeichnungen diskutiert werden, auch wenn der Ausgangspunkt dafür im Falle eines Computerprogrammes nur eine Fehlermeldung ist. Mithilfe von sogenannten Parser-Programmen kann nämlich der ausgezeichnete Text anhand der Regeln der DTD einer Überprüfung unterzogen werden, an deren Ende ein fehlerfreies, sprich valides, Dokument steht.

Wenn der Entschluß fest steht, eine DTD zu schreiben, unterwirft sich der Ersteller aber dem nächst höheren Regelwerk, nämlich der Metasprache, die die Erstellung der Regeln beschreibt. Im Moment kommt dafür zum einen SGML und zum anderen XML als Untermenge von SGML in Frage. Ich habe mich für XML als übergeordnetes Regelwerk entschieden aus Gründen, auf die ich im Weiteren noch näher eingehen möchte. Ob zuerst ausgezeichnet und dann über die Praxis ein praktikables Regelwerk gefunden wird oder zuerst die Regeln formuliert werden, ist meiner Meinung nach Geschmacksache. Ich habe zuerst ausgezeichnet und dann die Regeln schriftlich fixiert. Eine solche DTD für ein Diplom findet sich in Anhang 2.

Ein Auszeichnungselement wird durch das Schlüsselwort <!ELEMENT ...> definiert. Nach dem Schlüsselwort folgt der Name der Auszeichnung und das Inhaltsmodell. Im Inhaltsmodell wird beschrieben, was alles zwischen der Auszeichnung stehen darf (etwa Zeichen und / oder andere Auszeichnungen; gleichzeitig werden noch Aussagen über Anzahl und Reihenfolge bestimmter Inhalte getroffen).

1.4 Generierung

Obwohl durch Verweise auf externe Dateien die rein quantitative Anzahl der Auszeichnungen verringert werden kann, wird der ausgezeichnete Text im Prozeß der fortschreitenden Markierung zunehmend unübersichtlicher. Bedenken Sie, daß bei den hier vorgestellten Beispielen nur ein Bruchteil der denkbaren Auszeichnungen verwendet wurden. Es ist wichtig, zwischen der Speicherform und der Anwenderform zu unterscheiden. Der ausgezeichnete Text ist nur für die Speicherung da. Aus der Speicherform wird dann erst mittels zusätzlicher Programme die Anwenderform (Ausgabeform) erstellt. Deshalb wird kein Benutzer jemals eine spitze Klammer oder ein Hochkomma sehen. Auch der Editor kann bei der Bearbeitung der Dateien mittels geeigneter Software die Übersichtlichkeit verbessern, indem er bestimmte Auszeichnungsgruppen ausblendet oder sie durch farbige Markierungen hervorhebt.

Wie schon angedeutet, muß der ausgezeichnete Text unter Zuhilfenahme von selbstgeschriebenen Programmen (oder Stilvorlagen) in eine Form gebracht werden, die für einen Benutzer akzeptabel ist. Selbstgeschrieben müssen die Programme deshalb sein, da wahrscheinlich noch niemand vorher mit den gewählten Auszeichnungen gearbeitet hat. In dem vorliegenden Beispiel wurde nur eine einfache HTML Ausgabe realisiert.

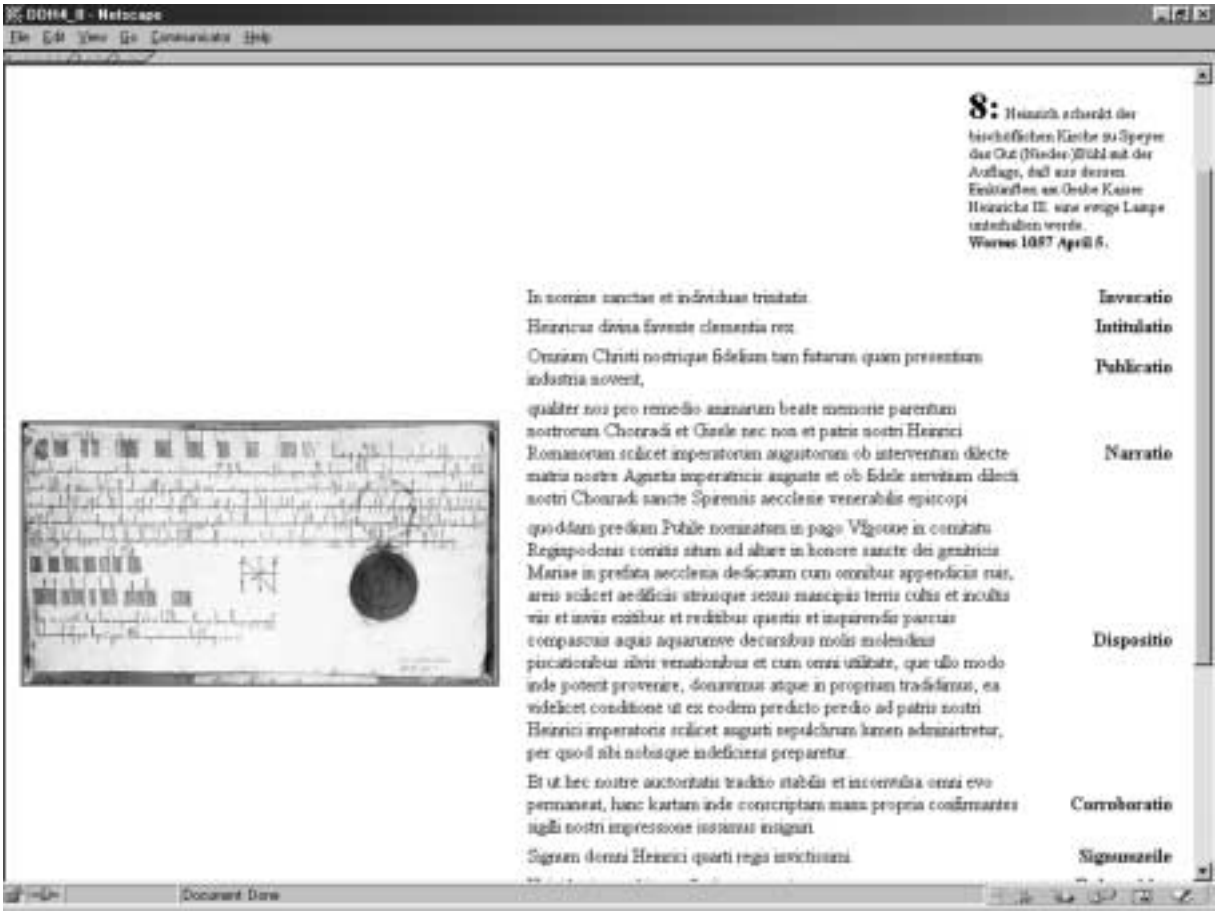
Es sind aber grundsätzlich vielfältige Möglichkeiten bei der Ausgabe gegeben. Als Programmiersprache fiel die Wahl auf Perl. Diese Programmiersprache eignet sich sehr gut für die Verarbeitung von reinen Textdateien. Es können aber auch andere Programmiersprachen benutzt werden, wie etwa ein Word Makro (das auf Visual Basic basiert). In meinem Fall habe ich mir die Programmiersprache selbst beigebracht, und wie Sie später noch am Ergebnis sehen können, hat es funktioniert. Das heißt jetzt nicht, daß ich ein guter Programmierer wäre, sondern eher, daß die Anforderungen, die an das Programmiergeschick gestellt wurden, gering waren. Grundsätzlich funktioniert das Programm nach dem Schema „Suche bestimmte Teile in der Speicherform, verändere sie nach bestimmten Vorgaben und schreibe die veränderten Teile in die Anwenderform". Bei der Generierung der Ausgabe tritt ein fundamentaler Unterschied in der Herangehensweise gegenüber der Auszeichnung auf. Denn die verwendeten Programme und das Ausgabemedium selbst brauchen in keinster Weise zukunftssicher zu sein. Ob nun Perl oder ein Word Makro verwendet wird, alle Lösungen werden den üblichen Lebenszyklen von Computertechnologie unterworfen sein. Aber weil die Anforderungen an die Programme so gering sind, kann dies verschmerzt werden, und es kann als sicher angenommen werden, daß zukünftig immer noch nach Textmustern gesucht und diese in veränderter Form gespeichert werden können. Die Prognose zu den hier verwendeten Systemen für die nächsten 50 Jahre ist: Perl hoffnungslos veraltet, HTML 4.0 gar nicht mehr bekannt, XML kann noch im Geschichtslexikon nachgeschlagen werden, aber die Speyrer Diplome Heinrichs IV.: immer noch von allen Rechnern lesbar und ein wahrer Quell an Wissen.

Leider besteht bei sehr vielen Projekten das Problem, daß die notwendigen EDV-Lösungen in einer Situation gestörter Kommunikation „gefunden" werden. Vertreter der historischen und der technischen Seite stehen sich in der Regel gegenüber, ohne über eine gemeinsame Sprache zu verfügen. Oft ist nämlich die Perspektive der Historiker: „Ich bin Historiker und diese ganzen technischen Dinge, das überlaß ich mal den Informatikern. Die werden das dann schon irgendwie richten." Aber dann passiert genau das. Sie richten es irgendwie und es funktioniert auch und hat bestimmt nicht viel der Zeit der Historiker und auch nur wenig Geld gekostet, aber das Ergebnis wird den Historiker nicht befriedigen und auch nicht dessen Vorstellungen von langfristig nutzbaren Ergebnissen entsprechen. Der Informatiker bleibt eben ein Informatiker, und er weiß nicht, daß ein und derselbe Name im gleichen Dokument ohne Weiteres in fünf verschiedenen Schreibweisen vorkommen kann, geschweige denn, was eine Narratio ist, und daß es einen Unterschied macht, ob eine Person in der Narratio oder in der Rekognitionszeile vorkommt. Wenn hingegen der Historiker über die grundlegenden technischen Zusammenhänge wenigstens grob informiert ist, dann braucht er oder sie immer noch keine Programmiersprache zu können, aber der Informatiker wird genau das machen, was ihm vorgeschrieben wird. Darüber hinaus kann man sich mit ihm verständigen und entsprechende Anweisungen geben (etwa: lösen Sie das Problem unter Benutzung von Auszeichnungssprachen oder einer relationalen Datenbank. Dazu muß allerdings bekannt sein, daß es so etwas gibt). Das Ergebnis wird dann viel eher zu überzeugen wissen. Wer sich statt dessen weiter nur auf die informatischen Dienstleistungen verlassen möchte, denke einen Augenblick an all die mißglückten geschichtlichen CD-Roms,

die eigentlich nur dafür gut sind, den gesamten Text zu exportieren, als ASCII Datei zu speichern und wieder von vorne anzufangen.

1.5 Effizienz

Kommen wir zurück zu den Diplomen. Nachfolgend sehen Sie einen Screenshot der HTML Ausgabe[1].



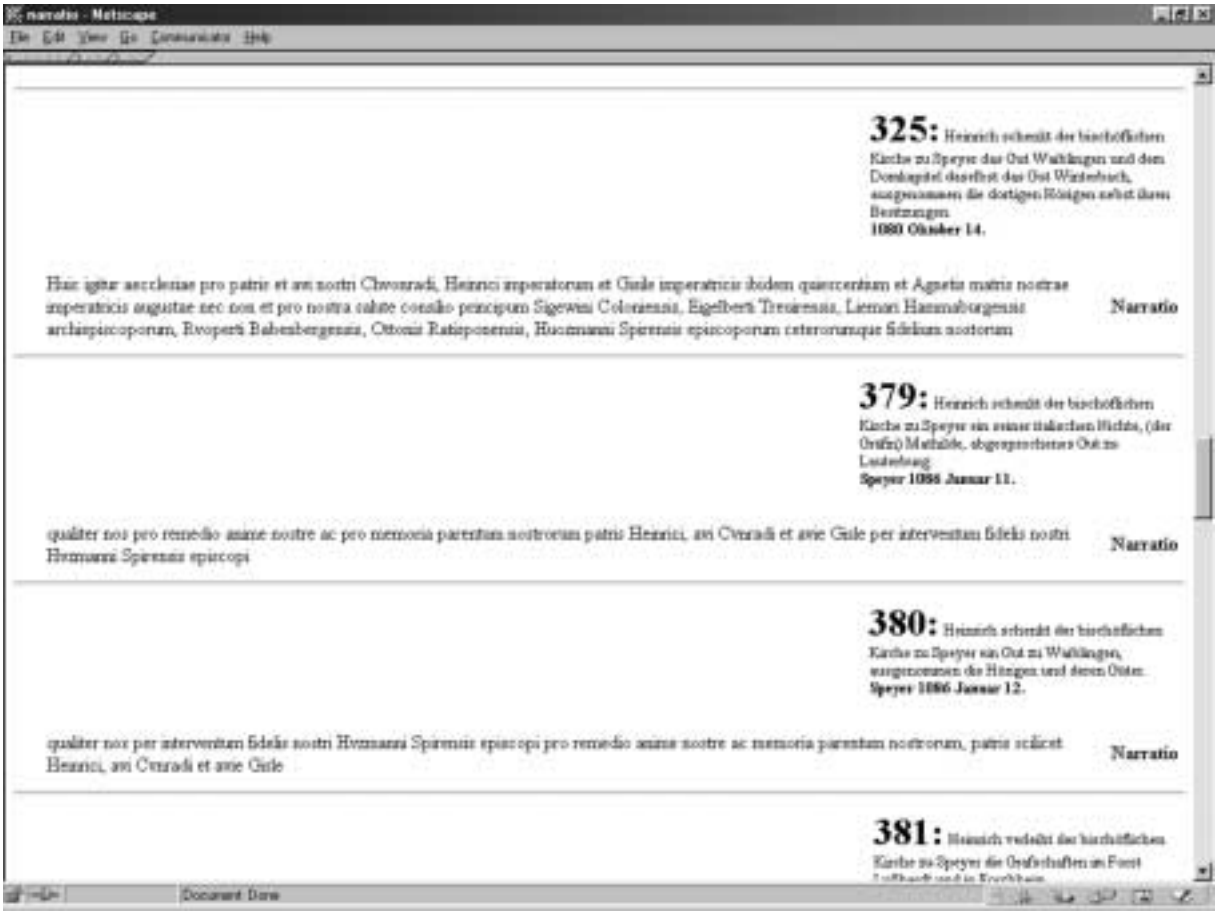
Diese Ausgabeform ist automatisch generiert worden und entspricht meinem persönlichen Layoutgeschmack. Wenn mir die Seitengestaltung nicht mehr gefallen sollte, oder mich ein Benutzer überzeugt hat, es sei zu häßlich, ändere ich fünf Zeilen in meinem Perl Programm, und schon kann alles ganz anders aussehen.

Durch die Methode der Textauszeichnung ist es möglich, personalisierte Ausgaben anzubieten. Doch dabei gilt der Grundsatz: was nicht ausgezeichnet wurde, steht auch nicht zur Verfügung. Da ich die Empfänger der Diplome nicht ausgezeichnet habe, was

1) Vgl. Online unter http://www.uni-koeln.de/~alh80/diplome/index.html. Der Screenshot zeigt das Diplom D8.

in meinem Fall auch keinen Sinn machte, kann ich auch keine nach den Empfängern geordnete Liste generieren. Genauer gesagt: wenn ich eine solche Liste benötigte, müßte ich sie von Hand erstellen, was nicht der Sinn der Sache ist.

Ich komme zu einem Beispiel für eine mögliche personalisierte Ausgabe, mit der effizienteres Arbeiten als mit der Buchausgabe möglich ist. Wenn jemand für seine Fragestellung z.B. nur die Narrationes der Diplome brauchte, wäre eine solche Auflistung



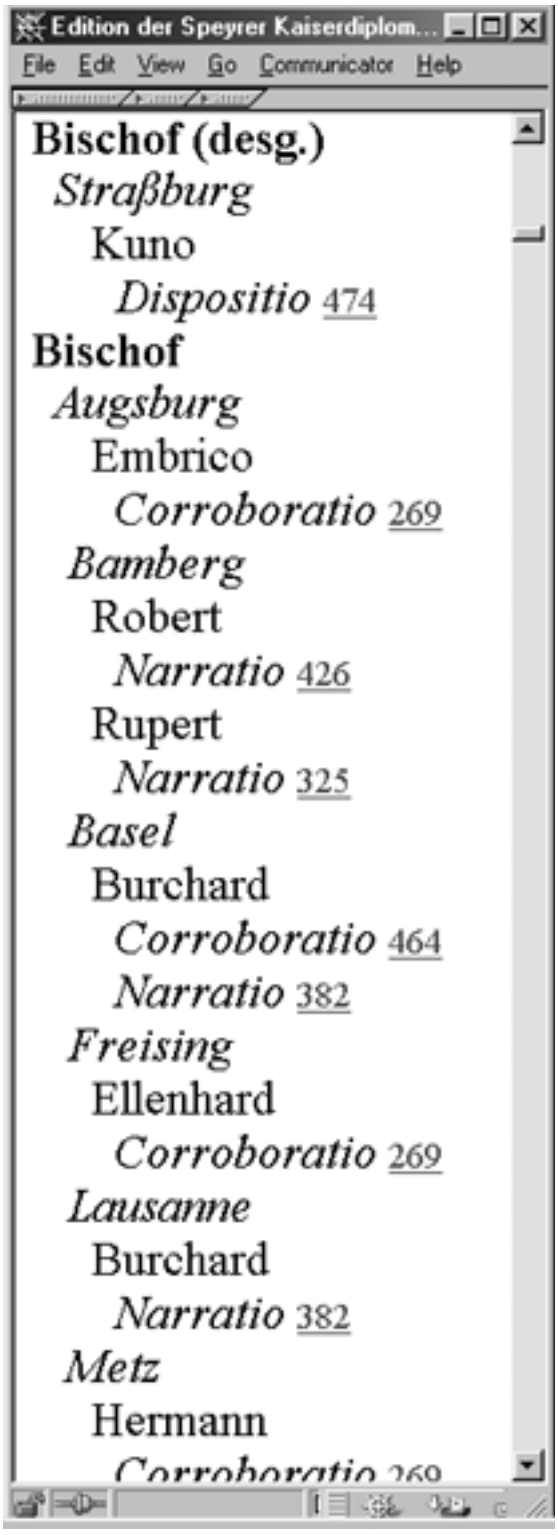
genau das, was er sich als Forscher von der Edition wünschen würde. Nachfolgend sehen Sie einen Screenshot einer automatisch generierten Narrationes-Liste.

Um noch besser mit solchen Texten arbeiten zu können, wäre eine Verknüpfung mit externen Wissensbasen, wie etwa lexikographischen Werken, von großem Nutzen. So sollte es beispielsweise zukünftig möglich sein, einen vorkommenden Personennamen mit der Online-Version des Lexikons des Mittelalters zu verknüpfen. Bis dies möglich ist, muß man sich anders behelfen.

1.6 Register

Wie aus der DTD ersichtlich ist, wurde bei der Auszeichnung der Namen und Orte das Attribut n (für Namen) verwendet. Der Inhalt dieses Attributes bildet die Grundlage für das Register. Im Projekt wird jede Person und jeder Ort auf drei verschiedene Arten

bestimmt. Erstens mit seinem Namen in heutiger Sprache, zweitens nach Amt, Landschaft und Herkunft und drittens nach seiner Art (also ob der Ort eine Stadt, ein Gau, ein Gut oder etwas anderes ist). Um das Register umfassend nutzen zu können, wurde jeder der ausgezeichneten Namen oder Orte auch nach allen drei Arten in das Register aufgenommen. Der Benutzer findet beispielsweise den Bischof Huzmann von Speyer unter den Stichworten *Bischof, Huzmann* und *Speyer*. Gleichzeitig merkt sich das Programm, welches das Register generiert, den Kontext, in dem der Eintrag gefunden wurde, hier also den Urkundenteil. Da das Register automatisch generiert wird, kann man sich bei der Arbeit ganz auf die Auszeichnung des Textes konzentrieren.



So sieht das Ergebnis aus:

Auch hier gilt: die Optik entspricht meinem Geschmack und kann leicht verändert werden. Mit dieser Art von Register ist ebenfalls ein effizienteres Arbeiten möglich, da auch der Urkundenteil angezeigt wird, in dem die Person vorkommt.

2 XML

XML (Extensible Markup Language) ist eine Metasprache in der die Regeln beschrieben sind, die zur Schaffung von Auszeichnungssprachen nötig sind. Sie stellt eine Vereinfachung von SGML dar und wurde speziell für das WWW geschaffen. Durch die Vereinfachung ist es für die Softwareindustrie leichter möglich, Programme anzubieten, die mit ausgezeichneten Texten umgehen können. Es gibt heute bereits rund 15 XML-Browser und eine Vielzahl sonstiger Programme rund um XML. Im Gegensatz dazu existieren für SGML nicht annähernd so viele Programme und falls doch, nutzen sie nicht alle Möglichkeiten, die SGML grundsätzlich bereit hält. SGML ist als Regelwerk zu mächtig, aber daraus resultierend auch zu kompliziert. Mein Eindruck ist, daß die professionellen Weblösungen zukünftig auf XML setzen werden, da sich mit XML leicht große Datenbanken aufbauen lassen. Mittels geeigneter Programme wird dem Benutzer dann wieder der Teil in einer Anwenderform geboten, der ihn besonders interessiert. Es ist festzustellen,

daß die Softwareindustrie erkannt hat, daß ein Bedarf an Programmen besteht, die mit XML-Dateien umgehen können. Dies bedeutet, daß Projekte, die mit Auszeichnungen arbeiten, mit der leichten Verfügbarkeit geeigneter Programme rechnen können.

2.1 XSL

Da ein XML Dokument nur Inhalt und Struktur besitzt und keinerlei Layoutangaben, braucht man für eine vernünftige Darstellung ein sogenanntes Stylesheet. Bei XML ist dafür die Extensible Stylesheet Language (XSL) zuständig. XSL ist noch nicht fertig gestellt. Damit aber schneller Ergebnisse präsentiert werden konnten, entschloß sich das World Wide Web Consortium (W3C)[2], die einzelnen Elemente von XSL von einander zu trennen. Ein Teil davon ist nun XSLT (das T steht für Transformationen) und ein anderer sind die Flow-Objects (Blockelemente). Mit XSLT können XML-Dateien in ein anderes Format transformiert werden. Es kann also der Schritt eingespart werden, ein Programm zu schreiben, das die HTML-Ausgabe generiert. Unter Zuhilfenahme der Suchsprache für XML-Strukturen (XPath) werden die ausgezeichneten Dateien dann gemäß den Angaben bearbeitet. Die Flow-Objects übernehmen jenen Teil, der schon durch die Cascading Style Sheets (CSS) bekannt ist. Bei den Flow-Objects ist noch vieles im Fluß. Es bleibt abzuwarten, was in diesem Bereich in nächster Zukunft passieren wird.

Für die Transformation wurde im Projekt der frei verfügbare XSLT Prozessor Xalan[3] verwendet. Der Prozessor ist eine implementiert die Spezifikation des W3C. Die folgenden Stylesheets machen im Grunde nichts anderes als das Perl-Skript. Deshalb sehen die daraus resultierenden HTML-Dateien genau so aus, wie die Dateien, die mit dem Perl-Skript erstellt wurden. Das Stylesheet gliedert sich in seiner Logik in zwei Blöcke. Zunächst wird bestimmt, aus welchen Teilen die HTML Datei besteht. Als zweiter größerer Block schließen sich Bestimmungen für einzelne Auszeichnungen an. Es wird also zuerst das „Was wird angezeigt?“ und dann das „Wie soll das ganze dann aussehen?“ bestimmt.

Da die Flow-Objects noch nicht endgültig definiert sind, muß hier der Umweg über HTML gegangen werden. HTML wird in diesem Zusammenhang nur für das Layout benutzt. Gleichzeitig könnten aber noch mehrere andere Layouts unterstützt werden, beispielsweise für LaTeX (professionelles Satzsystem), oder PDF bzw. einfachen ASCII Text. Im Bereich der Online-Publikation wäre es natürlich wünschenswert, wenn ein Anwendungsprogramm (z.B. ein WWW-Browser) die XML-Datei mitsamt dem zugehörigen Stylesheet sofort interpretieren könnte. Erste Ansätze dafür existieren

2) Das W3C definiert in Arbeitsgruppen die Standards für diverse Auszeichnungssprachen (unter anderem HTML und XML) und vieles mehr, vgl. auch Online unter http://www.w3c.org/.

3) Vgl. Online unter http://xml.apache.org/.

schon. So kann der Internet Explorer (IE) ab der Version 5 mit XSL und XSLT-Dateien umgehen. Die Stylesheets finden sich im Anhang, ab Beispiel 3.

3 Urkunden-DTD

Schon vor diesem Projekt zu Heinrich IV. hat sich mindestens noch eine weitere Person mit mittelalterlichen Urkunden und SGML/XML beschäftigt: Frau Annegret Fiebig (Tübingen) im Rahmen ihrer Dissertation[4]. Beide Projekte verbindet die jeweils eigene, für die unterschiedlichen Bedingungen angepaßte DTD und eigene Programme, die für die Ausgabe zuständig waren. Möglicherweise gibt es noch andere, die sich auf diese Art mit Urkunden beschäftigt haben oder sich beschäftigen werden. Für zukünftige Projekte wäre wünschenswert, daß sich ein gemeinsamer Standard (also eine DTD) durchsetzen würde. Dieser müßte noch entwickelt werden, denn die hier vorgestellte – exemplarisch ausgerichtete - DTD kann für ein größeres Projekt nicht verwendet werden. Die Vorteile eines gemeinsamen Standards liegen auf der Hand. Es gäbe nur eine DTD, und der Bearbeiter eines neues Projektes bräuchte sich nicht erst Gedanken über die Auszeichnungen zu machen. Durch eine Vereinheitlichung könnte auch erreicht werden, daß Programme entwickelt werden, die speziell für die Auszeichnung von Urkunden ausgelegt sind. Das fängt bei einem speziellen Eingabeprogramm an und endet bei der Generierung der Ausgaben. Ein Vergleich mit der Entwicklung von HTML zeigt, wie groß das Softwareangebot sein kann, wenn sich ein gemeinsamer Standard durchsetzt. Eine Definition in XML bietet sich im Moment an, und wäre auch leicht zu realisieren. Gleichzeitig wäre es wünschenswert, wenn sich für Urkunden oder für Quellen überhaupt eine Institution entwickeln würde wie das W3C für das WWW, die eine kontinuierliche Pflege des Standards gewährleisten könnte. Dafür in Frage käme beispielsweise eine Institution, die sich schon über ein Jahrhundert lang um mittelalterliche Urkunden verdient gemacht hat. Dort würden Arbeitsgruppen Vorschläge für eine DTD entwickeln und zur Diskussion freigeben. Nach einer angemessenen Zeit würde der Vorschlag dann überarbeitet und als Empfehlung veröffentlicht. An dieser Empfehlung wiederum orientierten sich schließlich die Bearbeiter (die Historiker) und die von Seiten der Informatik zu entwikkelnden Werkzeuge.

Als abschließendes Fazit ist zu formulieren: die technischen Voraussetzungen für die Auszeichnung großer Urkundenbestände sind gegeben; es ist mit kalkulierbarem Aufwand zu realisieren; was fehlt, sind vor allem eine einheitliche DTD und die darauf aufbauende Hauptsache: ausgezeichnete Urkundentexte.

---

4) Die Dissertation lag zum Zeitpunkt des Vortrages noch nicht im Druck vor.

4. Anhang

Bsp. 1: das ausgezeichnete Diplom D8.

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!--
Nach der XML Deklaration folgt nun die Angabe der Dokument Typ Definition
( DTD, wird gleich noch erklärt ).
-->
<!DOCTYPE urkunde SYSTEM „urkunden.dtd">
<!--
Hier nun die erste Auszeichnung <urkunde>. Alles was in spitzen Klammern
steht, ist eine Auszeichnung. Zu jeder Auszeichnung gehört einmal ein
öffnendes ( <urkunde> ) und ein schließendes Element ( </urkunde>, am Ende der
Datei ). Als zusätzliche Regel gilt, daß sich die Auszeichnungen nicht
überlappen dürfen ( also nicht: <urkunde><vorspann></urkunde></vorspann> ).
-->
<urkunde>
<vorspann>
<bild>8.jpg</bild>
<nr>8</nr>
<!--
Der Name dieser Auszeichnung ( <regest> ) ist von mir frei gewählt worden und
beinhaltet das Wissen um diesen Textabschnitt. Natürlich weiß auch jeder
Benutzer der Edition in Buchform, daß dieser Abschnitt das Regest ist, da
es als eigener Absatz in kursiver Schrift gedruckt worden ist. Diese
Informationen gehen aber wie hier zu sehen ist, verloren. Bei
Auszeichnungssprachen ist die Struktur des Textes nicht mit dem Layout
verwoben. Aus diesem Grund heißt die Auszeichnung auch <regest> und nicht
<Absatz_in_kursiver_Schrift>. Ein weiterer Vorteil ist, daß es möglich ist,
diese Datei durch einen Rechner effizient weiterzuverarbeiten. Die
Weiterverarbeitung beschränkt sich nicht nur auf einfache Suche nach
Zeichenketten, sondern es können mächtigere Verfahren zum Einsatz kommen.
-->
<regest>Heinrich schenkt der bischöflichen Kirche zu Speyer das Gut
(Nieder-)Bühl mit der Auflage, daß aus dessen Einkünften am Grabe Kaiser
Heinrichs III. eine ewige Lampe unterhalten werde.</regest>
<datum>Worms 1057 April 5.</datum>
</vorspann>
<text>
<invocatio>In nomine sanctae et individuae trinitatis.</invocatio>
<intitulatio>Heinricus divina favente clementia rex. </intitulatio>
<publicatio>Omnium Christi nostrique fidelium tam futurum quam presentium
industria noverit, </publicatio>
<narratio>qualiter nos pro remedio animarum beate memorie parentum
```

```
nostrorum <name nr="n8-1" n="Konrad II.:Kaiser:NN">Chonradi</name> et
<name
nr="n8-2" n="Gisela:Kaiserin:NN">Gisele</name> nec non et patris nostri
<name nr="n8-3" n="Heinrich III.:Kaiser:NN">Heinrici</name> Romanorum
scilicet imperatorum augustorum ob interventum dilecte matris nostre <name
nr="n8-4" n="Agnes von Poitu:Kaiserin:Poitu">Agnetis</name> imperatricis
auguste et ob fidele servitium dilecti nostri <name nr="n8-5"
n="Konrad:Bischof:Speyer">Chonradi</name> sancte Spirensis aecclesie
venerabilis episcopi </narratio>
<dispositio>quoddam predium <platz nr="p8-1"
n="Niederbühl:Kr. Rastatt:Gut">Puhile</platz> nominatum in pago <platz
nr="p8-2" n="Ufgau ö. des Rheins:Baden-Baden:Gau">Vfgouue</platz> in
comitatu <name nr="n8-6" n="Reginpodo:Graf:Ufgau ö. des
Rheins">Reginpodonis</name> comitis situm ad altare in honore sancte dei
genitricis <platz nr="p8-3" n="Hl. Maria:Altar:Dom zu
Speyer">Mariae</platz> in prefata aecclesia dedicatum cum omnibus
appendiciis suis, areis scilicet aedificiis utriusque sexus mancipiis
terris cultis et incultis viis et inviis exitibus et reditibus questis et
inquirendis pascuis compascuis aquis aquarumve decursibus molis molendinis
piscationibus silvis venationibus et cum omni utilitate, que ullo modo inde
poterit provenire, donavimus atque in proprium tradidimus, ea videlicet
conditione ut ex eodem predicto predio ad patris nostri <name nr="n8-7"
n="Heinrich III.:Kaiser:NN">Heinrici</name> imperatoris scilicet augusti
sepulchrum lumen administretur, per quod sibi nobisque indeficiens
preparetur. </dispositio>
<corroboratio>Et ut hec nostre auctoritatis traditio stabilis et inconvulsa
omni evo permaneat, hanc kartam inde conscriptam manu propria confirmantes
sigilli nostri impressione iussimus insigniri. </corroboratio>
<signum>Signum domni Heinrici quarti regis invictissimi.</signum>
<rekognition><name nr="n8-8" n="Winither:Kanzler:NN">Uuinitherius</name>
cancellarius vice <name nr="n8-9"
n="Liutbald:Erzkanzler:Mainz">Liutpaldi</name> archicancellarii
recognovi.</rekognition>
<datierung>Datas nonas apr. anno dominice incarnationis MLVII, indictione
X, anno autem domni Heinrici quarti regis ordinationis eius tertio, regni
vero primo; actum <platz nr="p8-4" n="Worms:Rheinhessen
Pfalz:Stadt">Wormatie</platz>; </datierung>
<apprecatio>in nomine domini feliciter amen.</apprecatio>
</text>
</urkunde>
```

Bsp. 2: die zugrungeliegende DTD des Projektes:

```
<!-- Urkunden DTD für die Diplome HIV. für Speyer -->
<!-- von Bernhard Assmann -->
<!-- Benutzung: <!DOCTYPE urkunde SYSTEM „urkunde.dtd“> -->
<!ENTITY % reg „name | platz“>
<!ENTITY % format „vu | luecke“>
<!ENTITY % echt „unecht | vunecht“>
<!ELEMENT urkunde (vorspann,text)>
<!ELEMENT vorspann (bild*,nr,regest,datum)>
<!ELEMENT text ((invocatio | intitulatio | inscriptio | arenga |
        publicatio | narratio | dispositio | corroboratio |
        sanctio | signum | rekognition |
        datierung | apprecatio)*)>
<!ELEMENT bild (#PCDATA)>
<!ELEMENT nr (#PCDATA | %echt;)*>
<!ELEMENT regest (#PCDATA | %echt;)*>
<!ELEMENT datum (#PCDATA | %echt;)*>
<!ELEMENT invocatio (#PCDATA | %echt; | %format; | %reg;)*> <!-- eine Invocatio -->
<!ELEMENT intitulatio (#PCDATA | %echt; | %format; | %reg;)*>
<!ELEMENT inscriptio (#PCDATA | %echt; | %format; | %reg;)*>
<!ELEMENT arenga (#PCDATA | %echt; | %format; | %reg;)*>
<!ELEMENT publicatio (#PCDATA | %echt; | %format; | %reg;)*>
<!ELEMENT narratio (#PCDATA | %echt; | %format; | %reg;)*>
<!ELEMENT dispositio (#PCDATA | %echt; | %format; | %reg;)*>
<!ELEMENT corroboratio (#PCDATA | %echt; | %format; | %reg;)*>
<!ELEMENT sanctio (#PCDATA | %echt; | %format; | %reg;)*>
<!ELEMENT signum (#PCDATA | %echt; | %format; | %reg;)*>
<!ELEMENT rekognition (#PCDATA | %echt; | %format; | %reg;)*>
<!ELEMENT datierung (#PCDATA | %echt; | %format; | %reg;)*>
<!ELEMENT apprecatio (#PCDATA | %echt; | %format; | %reg;)*>
<!ELEMENT vu (#PCDATA | %echt; | %format; | %reg;)*>
<!ELEMENT luecke EMPTY>
<!ELEMENT unecht (#PCDATA | %echt; | %format; | %reg;)*>
<!ELEMENT vunecht (#PCDATA | %echt; | %format; | %reg;)*>
<!ELEMENT name (#PCDATA)>
<!ELEMENT platz (#PCDATA)>
<!ATTLIST name
        nr ID #REQUIRED
        n CDATA #IMPLIED>
<!ATTLIST platz
        nr ID #REQUIRED
        n CDATA #IMPLIED>
```

Bsp. 3: das XSL-Stylesheet für ein Diplom

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!--
Das Stylesheet beginnt mit der Angabe des Namensraums. Hier wird der
korrekte Namensraum für XSL verwendet. Alle folgenden Auszeichnungen die
mit xsl: beginnen, beziehen sich auf den XSL-Namensraum, alle anderen
Auszeichnungen auf HTML.
-->
<xsl:stylesheet version="1.0"
            xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<!--
Angaben für den XSLT Prozessor. Es soll eine HTML Datei erzeugt werden, mit
Einrückungen in dem hier üblichen Zeichensatz.
-->
<xsl:output method="html"
          indent="yes"
          encoding="iso-8859-1"/>
<!--
Definition einiger Variabeln, die später verwendet werden. Der Name der
Variabeln befindet sich in dem Attribut name, der Wert zwischen den
Auszeichnungen.
-->
<xsl:variable name="keywords">Diplomatik Diplome Urkunde Edition XML DTD
GEDEQ</xsl:variable>
<xsl:variable name="email">b.assmann@uni-koeln.de</xsl:variable>
<xsl:variable name="name">Bernhard Assmann</xsl:variable>
<!--
Der erste Verarbeitungsschritt. Aus dem XML Baum wird der Ast <urkunde>
ausgeschnitten.
-->
<xsl:template match="urkunde">
<!--
Dies ist ein kleiner Trick, um die DOCTYPE Zeile vernünftig
hinzubekommen. Gleichzeitig ist es auch schon die erste Zeile, die vom XSLT
Prozessor ausgegeben wird.
-->
<xsl:text disable-output-escaping="yes">
<![CDATA[
<!DOCTYPE html PUBLIC „-//W3C//DTD HTML 4.01 Transitional//EN">
]]>
</xsl:text>
<!--
Der Beginn einer HTML Datei. Es werden die ersten Variabeln eingesetzt,
```

```
z.B. {$email}
-->
<html>
<head>
<link href="mailto:{$email}" rev="made" />
<link href="mailto:{$email}" rev="owns" title="{$name}" />
<meta http-equiv="content-type" content="text/html;CHARSET=iso-8859-1" />
<meta name="Author" content="{$name}" />
<meta name="keywords" content="{$keywords}" />
<!--
Erzeugung der Titelzeile im Browser. Mit der Angabe value-of wird der
Inhalt einer Auszeichnung eingefügt, die im select Attribut näher
beschrieben ist. Der innere Aufbau des select Attributs entspricht der
XPATH Spezifikation und bedeutet hier: suche im Element vorspann das
Unterelement nr. Bedenken Sie, daß wir uns im Ast urkunde des XML Baums
befinden. Eventuell eingebettete Auszeichnung gehen verloren.
-->
<title>DD HIV. Nr. <xsl:value-of select="vorspann/nr"/></title>
</head>
<body bgcolor="white" text="black" link="blue" vlink="purple" alink="aqua">
<h1 align="center">DHIV. <xsl:value-of select="vorspann/nr"/></h1>
<p><a href="">Zurück zum Auswahlmenü</a></p>
<font color="fuchsia">Fuchsia</font> kennzeichnet Text aus
einer Vorurkunde (Kleindruck in den MGH),<br/>
<font color="green">Grün</font> kennzeichnet verunechtete Diplome und<br/>
<font color="red">Rot</font> kennzeichnet unechte Diplome.<br/>
<hr/>
<!--
Hier wird mit apply-templates der Inhalt eingefügt und eventuell
existierende Unterelemente bleiben erhalten ( unecht, vunecht usw. )
-->
<table><tr><td width="80%"></td><td width="20%" align="left">
<font size="+3"><b><xsl:apply-templates select="vorspann/nr"/>:</b></font>
<font size="-1">
<xsl:apply-templates select="vorspann/regest"/><br/><b>
<xsl:apply-templates select="vorspann/datum"/>
</b></font></td></tr>
</table>
<table>
<!--
Es sind auch Bedingungen möglich: es wird im folgenden für jeden
Urkundenteil abgefragt, ob er den überhaupt vorhanden ist, und erst dann
wird die entsprechende Tabellenzeile erzeugt. Die Bedingung steckt im
Attribut test von when und bedeutet: wenn irgendwo eine Auszeichnung mit
```

```
dem Namen invocatio, dann führe alles bis /xsl:when aus.
-->
<xsl:choose>
<xsl:when test="//invocatio">
<tr><td width="15"></td><td width="90%">
<xsl:apply-templates select="text/invocatio"/>
</td><td width="10%" align="right"><b>Invocatio</b></td></tr>
</xsl:when>
</xsl:choose>
<xsl:choose>
<xsl:when test="//intitulatio">
<tr><td width="15"></td><td width="90%">
<xsl:apply-templates select="text/intitulatio"/>
</td><td width="10%" align="right"><b>Intitulatio</b></td></tr>
</xsl:when>
</xsl:choose>
<xsl:choose>
<xsl:when test="//publicatio">
<tr><td width="15"></td><td width="90%">
<xsl:apply-templates select="text/publicatio"/>
</td><td width="10%" align="right"><b>Publicatio</b></td></tr>
</xsl:when>
</xsl:choose>
<xsl:choose>
<xsl:when test="//narratio">
<tr><td width="15"></td><td width="90%">
<xsl:apply-templates select="text/narratio"/>
</td><td width="10%" align="right"><b>Narratio</b></td></tr>
</xsl:when>
</xsl:choose>
<xsl:choose>
<xsl:when test="//dispositio">
<tr><td width="15"></td><td width="90%">
<xsl:apply-templates select="text/dispositio"/>
</td><td width="10%" align="right"><b>Dispositio</b></td></tr>
</xsl:when>
</xsl:choose>
<xsl:choose>
<xsl:when test="//corroboratio">
<tr><td width="15"></td><td width="90%">
<xsl:apply-templates select="text/corroboratio"/>
</td><td width="10%" align="right"><b>Corrobaratio</b></td></tr>
</xsl:when>
</xsl:choose>
```

```
<xsl:choose>
<xsl:when test="//signum">
<tr><td width="15"></td><td width="90%">
<xsl:apply-templates select="text/signum"/>
</td><td width="10%" align="right"><b>Signumzeile</b></td></tr>
</xsl:when>
</xsl:choose>
<xsl:choose>
<xsl:when test="//rekognition">
<tr><td width="15"></td><td width="90%">
<xsl:apply-templates select="text/rekognition"/>
</td><td width="10%" align="right"><b>Rekognition</b></td></tr>
</xsl:when>
</xsl:choose>
<xsl:choose>
<xsl:when test="//datierung">
<tr><td width="15"></td><td width="90%">
<xsl:apply-templates select="text/datierung"/>
</td><td width="10%" align="right"><b>Datierung</b></td></tr>
</xsl:when>
</xsl:choose>
<xsl:choose>
<xsl:when test="//apprecatio">
<tr><td width="15"></td><td width="90%">
<xsl:apply-templates select="text/apprecatio"/>
</td><td width="10%" align="right"><b>Apprecatio</b></td></tr>
</xsl:when>
</xsl:choose>
</table>
<hr width="100%"/>
<p><font color="fuchsia">Fuchsia</font> kennzeichnet Text aus
einer Vorurkunde (Kleindruck in den MGH),<br/>
<font color="green">Grün</font> kennzeichnet verunechtete Diplome und<br/>
<font color="red">Rot</font> kennzeichnet unechte Diplome.<br/></p>
<p><a href="">Zurück zum Auswahlmenü</a></p>
<address><a href="mailto:{$email}"><xsl:value-of select="$name"/>,</a>,
HTML Version © <xsl:value-of select="$name"/>.</address>
<p><a href="http://validator.w3.org/check/referer"><img border="0"
src="valid-html401" alt="Valid HTML 4.01!" height="31" width="88"/></a></p>
</body>
</html>
</xsl:template>
<!--
Die HTML Datei ist vollständig angelegt worden. Jetzt folgen noch
```

Anweisungen zu einzelnen Auszeichnungen. Für jede Auszeichnung wird ein template verwendet und mit apply-templates der Inhalt eingefügt. Die folgenden templates färben Teile des Textes ein.
-->
<xsl:template match="vu"><xsl:apply-templates />
</xsl:template>
<xsl:template match="unecht">
<xsl:apply-templates />
</xsl:template>
<xsl:template match="vunecht">
<xsl:apply-templates />
</xsl:template>
</xsl:stylesheet>

Bsp. 4: das XSL-Stylesheet für die Narrationes:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0"
            xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html"
          indent="yes"
          encoding="iso-8859-1"/>
<xsl:variable name="keywords">Diplomatik Diplome Urkunde Edition XML DTD GEDEQ</xsl:variable>
<xsl:variable name="email">b.assmann@uni-koeln.de</xsl:variable>
<xsl:variable name="name">Bernhard Assmann</xsl:variable>
<xsl:template match="/">
<xsl:text disable-output-escaping="yes">
<![CDATA[
<!DOCTYPE html PUBLIC „-//W3C//DTD HTML 4.01 Transitional//EN">
]]>
</xsl:text>
<html>
<head>
<link href="mailto:{$email}" rev="made" />
<link href="mailto:{$email}" rev="owns" title="{$name}" />
<meta http-equiv="content-type" content="text/html;CHARSET=iso-8859-1" />
<meta name="Author" content="{$name}" />
<meta name="keywords" content="{$keywords}" />
  <title>Narrationes</title>
</head>
<body bgcolor="white" text="black" link="blue" vlink="purple" alink="aqua">
<h1 align="center">Narrationes</h1>
```

```
<p><a href="index2.html">Zurück zum Auswahlmenü</a></p>
<p><font color="fuchsia">Fuchsia</font> kennzeichnet Text aus
einer Vorurkunde (Kleindruck in den MGH),<br/>
<font color="green">Grün</font> kennzeichnet verunechtete Diplome und<br/>
<font color="red">Rot</font> kennzeichnet unechte Diplome.</p>
<!--
Der Trick hier ist, mit einer Hilfsdatei auf alle 32 Dateien
zuzugreifen. In der Datei dateien.xml stehen jeweils in der Auszeichnung
<datei> alle Dateinamen. Jetzt wird mit der for-each Anweisung jede
dieser Auszeichnungen nacheinander verarbeitet.
-->
<xsl:for-each select="dateien/datei/text()">
<hr width="50%"/>
<!--
Mit der XSLT Funktion document wird nun die erste Datei geöffnet und der
Inhalt von urkunde/vorspann/nr eingesetzt. Der Dateiname steckt in
string(.).
-->
<table><tr><td width="80%"></td><td width="20%" align="left">
<font size="+3">
<b>
<xsl:apply-templates select="document(string(.))//urkunde/vorspann/nr"/>
:</b>
</font>
<font size="-1">
 <xsl:apply-templates select="document(string(.))//urkunde/vorspann/regest"/
><br/>
            <b><xsl:apply-templates   select="document(string(.))//urkunde/vorspann/
datum"/>
</b></font></td></tr>
</table>
<table>
<tr><td width="15"></td><td width="90%">
<xsl:apply-templates select="document(string(.))//urkunde/text/narratio"/>
</td><td width="10%" align="right"><b>Narratio</b></td></tr>
</table>

</xsl:for-each>
<hr width="100%"/>
<p><font color="fuchsia">Fuchsia</font> kennzeichnet Text aus
einer Vorurkunde (Kleindruck in den MGH),<br/>
<font color="green">Grün</font> kennzeichnet verunechtete Diplome und<br/>
<font color="red">Rot</font> kennzeichnet unechte Diplome.<br/></p>
<p><a href="index2.html">Zurück zum Auswahlmenü</a></p>
```

```
<address><a href="mailto:{$email}"><xsl:value-of select="$name"/>,</a>,
HTML Version © <xsl:value-of select="$name"/>.</address>
<p><a href="http://validator.w3.org/check/referer"><img border="0"
src="valid-html401" alt="Valid HTML 4.01!" height="31" width="88"/></a></p>
</body>
</html>
</xsl:template>
<xsl:template match="vu"><font style="color:fuchsia"><xsl:apply-templates /></
font>
</xsl:template>
<xsl:template match="unecht">
<font style="color:red"><xsl:apply-templates /></font>
</xsl:template>
<xsl:template match="vunecht">
<font style="color:green"><xsl:apply-templates /></font>
</xsl:template>
</xsl:stylesheet>
```

Bsp. 5: die Hilfsdatei „dateien.xml" für das Narrationes-Stylesheet:

```
<?xml version="1.0" ?>
<dateien>
<datei>DDH4_008.xml</datei>
<datei>DDH4_009.xml</datei>
<datei>DDH4_010.xml</datei>
<datei>DDH4_011.xml</datei>
<datei>DDH4_012.xml</datei>
<datei>DDH4_028.xml</datei>
<datei>DDH4_044.xml</datei>
<datei>DDH4_078.xml</datei>
<datei>DDH4_087.xml</datei>
<datei>DDH4_100.xml</datei>
<datei>DDH4_165.xml</datei>
<datei>DDH4_166.xml</datei>
<datei>DDH4_167.xml</datei>
<datei>DDH4_269.xml</datei>
<datei>DDH4_277.xml</datei>
<datei>DDH4_325.xml</datei>
<datei>DDH4_379.xml</datei>
<datei>DDH4_380.xml</datei>
<datei>DDH4_381.xml</datei>
<datei>DDH4_382.xml</datei>
<datei>DDH4_383.xml</datei>
<datei>DDH4_384.xml</datei>
```

```
<datei>DDH4_385.xml</datei>
<datei>DDH4_391.xml</datei>
<datei>DDH4_396.xml</datei>
<datei>DDH4_426.xml</datei>
<datei>DDH4_464.xml</datei>
<datei>DDH4_466.xml</datei>
<datei>DDH4_474.xml</datei>
<datei>DDH4_475.xml</datei>
<datei>DDH4_480.xml</datei>
<datei>DDH4_489.xml</datei>
</dateien>
```

Bernhard Assmann,
Im Leimfeld 3, D - 51065 Köln, b.assmann@uni-koeln.de