# 7363 - Web-basierte Anwendungen:
# Meilenstein 1

Umgang mit DocBook
am Beispiel eines c't-Artikels



## Etappe 1

- (Die Anweisungen für Etappe 1 erfolgten mündlich)
- hier ausgelassen; bei Gelegenheit nachtragen bzw. die Folien aus dem 1. Vorlesungsteil hier integrieren.

# Abschluss der ersten Etappe

Prüfung der verwendeten Elemente
FO- und PDF-Erzeugung
Alternative HTML-Erzeugung



## Checkliste

- articleinfo
  - Titel, Untertitel, Abstract?
  - Autor, "authorblurb", e-mail, Copyright?
- Abbildungen
  - Begleittext ("caption")?
  - zentriert?
  - spezifiziert (Typ)?
- Tabellen
  - Kopfzeile?
  - Begleittext?
- Programm-Listing
  - Begleittext?
  - 1:1-Übernahme des Beispieltextes?!
- Literaturverzeichnis (2 Einträge)?

## Checkliste

- Block-Details
  - Beispiel-Codes: informalexample bzw. example genutzt?
- Inline-Details
  - filename vs. command vs. application: Klar unterscheiden!
  - quote, acronym, abbrev, emphasis: Konsequent eingesetzt?
  - Verwenden Sie userinput, envar, option, parameter? Auch korrekt?
  - Bei Beispieleingaben: replaceable verwendet?
  - Spezielle Tasten: Mit keycap, keycombo gearbeitet?



## FO-Ausgabe und PDF-Erzeugung

- Erweitertes Makefile
  - Nehmen Sie nun das erweiterte Makefile (in den Vorlagen) anstelle des bisherigen in Betrieb
  - Bei "make all" entsteht vorübergehend eine Datei "unterbau.fo". Diese wird an den FO-Prozessor FOP übergeben und anschließend automatisch gelöscht.
  - FOP erzeugt die Datei "unterbau.pdf"
    - Die zahlreichen Fehlermeldungen / Warnungen ignorieren Sie...

## FO-Ausgabe und PDF-Erzeugung

- Sichten Sie das Ergebnis, z.B. mit `acroread`.
  - Ist das Ergebnis vollständig? Optisch überzeugend?
  - Bewirken die Inline-Markups noch differenziertere Darstellungen als bei HTML? Das sollten sie!
  - Funktioniert die (deutsche) Wörtertrennung?
  - Wie sind die Tabellen gelungen?
  - Funktioniert Ihre Version der Literaturangaben auch hier?
- Bewerten Sie das Ergebnis kritisch.
  - Selbsttest: Würde ich auf diesem Weg meine Diplomarbeit erstellen?
  - Begründen Sie Ihre Bewertung.



## HTML-Ausgabe mit "chunking"

- Tauschen Sie im Makefile "docbook.xsl" gegen "chunk.xsl"
- Erzeugen Sie erneut den HTML-Output
- Wirkung?

# Etappe 2: Customizing des Outputs

Das Attribut "role"
Stylesheet-Parameter
Einsatz von CSS
Profiling
Custom stylesheets



## *abstraction* vs. *presentation*

- Das Standard-Attribut "role"
  - Bei Umwandlung in HTML wird der Wert von "role" oft auf ein "class"-Attribut gemappt.
  - Das lässt sich etwa via CSS zur gezielten Änderung der Darstellung verwenden.
    - Beispiel "para": Sie können unterschiedliche "para-Arten" schaffen
  - Sonderfall "emphasis"
    - Wie werden Texte in "emphasis" normalerweise in HTML dargestellt? Verwenden Sie emphasis in Ihren Artikel, notfalls nur zu Testzwecken.
    - Fügen Sie nun das Attribut
      role = "bold"
      zu Ihren emphasis-Elementen hinzu (mindestens zu einem).
    - Wirkung (ohne CSS)?
    - Wird das Prinzip "Trennung von Inhalt und Darstellung" beachtet?

# Stylesheet-Parameter

- Output-Steuerung
  - Es gibt **zahlreiche** Möglichkeiten, die Wirkung der Stylesheets zu beeinflussen.
  - Die wichtigsten (einfachsten?) Fälle sind über vordefinierte Stylesheet-Parameter vom Anwender auf einfache Weise veränderbar.
    - Zur Technik: Vgl. XSLT, Variablen und Parameter
  - Auch ganze Schablonenregeln lassen sich überladen (d.h. durch eigene ersetzen)!
- Dokumentation
  - Im jeweiligen Stylesheet-Verzeichnis (.../html, .../fo, etc.) befindet sich eine Docbook-Datei "param.xml"
  - Legen Sie ein Unterverzeichnis "params" analog zu "unterbau" an, kopieren Sie .../html/param.xml dorthin, erstellen Sie ein geeignetes Makefile und erzeugen Sie eine lesbare HTML-Dokumentation.



# Stylesheet-Parameter

- Parameter setzen
  - per Kommandozeile:
    - XSLT-Prozessoren besitzen Optionen, die es gestatten, Parameter und deren Werte an das Stylesheet "durchzureichen".
    - Beispiel Xalan: Option `-p` *`param_name param_wert`*
    - Beispiel xsltproc: Option `--stringparam` *`param_name param_wert`*
    - Nur in einfachen Fällen (wenige Parameter) praktikabel
  - per Custom-Script:
    - Einfach eintragen! Beispiele:
      `<xsl:param name="`*`param_name`*`">`*`param_wert`*`</xsl:param>`
      `<xsl:param name="`*`param_name`*`" select="`*`xpath_ausdruck`*`"/>`
    - Dies ist der Standardweg!
- Parameterauswahl
  - Die Dokumentation in params.xml enthält Name, Kurzbeschreibung und Voreinstellung zu den verfügbaren Parametern.
  - Im Zweifelsfall in den Quellen nachsehen (params.xsl)!

## Stylesheet-Parameter: Aufgaben

- FO-Ausgabe optimieren
  - Stellen Sie das Papierformat ein auf "A4"
  - Experimentieren Sie mit 2-spaltigem Textsatz! Ergebnis?
- HTML-Ausgabe variieren
  - Arbeiten Sie mit "chunking"
  - Aktivieren Sie die Navigations-Icons
    - Tipp: Legen Sie ein soft link "images" von Ihrem Arbeitsverzeichnis in das gleichnamige Unterverzeichnis im Stylesheet-Bereich!
  - Aktivieren Sie die Nutzung von "Wasserzeichen".
    - Verwenden Sie dazu das Hintergrundbild "draft.png" aus "images".
  - Legen Sie eine CSS-Datei "unterbau.css" an und veranlassen Sie deren Nutzung per Parameter
    - Einziger Eintrag: `.acronym { color: red }`
    - Was bewirkt dies? Wie funktioniert das?



## Custom Stylesheets

- Custom-Scripts:
  - Ein Custom-Script besteht typischerweise aus
    - dem üblichen Anfang eines XSLT-Scripts
    - einem "include"-Element, mit dem man das gewünschte Standard-Script einbindet (z.B. html/chunk.xsl)
    - "param"-Elementen zur Änderung von Voreinstellungen
    - "template"-Elementen zum Überschreiben der Standardversionen
  - Wirkung
    - Gemäß der allgemeinen XSLT-Regel, nach der die jeweils letzte Regel ihre Vorgänger überschreibt bzw. ersetzt, lassen sich durch Einfügen eigener Regeln hinter dem "include"-Element die Standardregeln beliebig ersetzen.
  - Anwendung
    - Einfach das Custom-Script anstelle des normalen in's Makefile eintragen!
- Merke:
  - Bei XSLT-Regeln "gewinnt" die letzte, bei Deklarationen in DTDs die erste!

# Custom Stylesheets: Beispiel

```
<xsl:stylesheet
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   version="1.0">
   <xsl:import href="styles/html/chunk.xsl"/>

  <xsl:param name="html.stylesheet">e-novative.css</xsl:param>

  <xsl:template name="user.header.content">
    <div id="customheader">
    <span class="logo">FH Wiesbaden</span>
    Fachbereich Informatik
    </div>
  </xsl:template>

  <xsl:template name="user.footer.content">
    <div id="customfooter">
      Copyright &#169; 2004 Fachhochschule Wiesbaden.
      All rights reserved.
     </div>
  </xsl:template>
</xsl:stylesheet>
```



# Custom Stylesheets: Beispiel

- Wirkung
  - HTML-Erzeugung, "chunking"-Variante
  - Definition einer CSS-Datei für die HTML-Ausgabe
  - Definition eigener Header- und Footer-Texte/Ausgaben
- Anwendung
  - Testen Sie die Wirkung dieses Stylesheets
  - Testen Sie ebenfalls die Wirkung der CSS-Datei
- Quelle
  - Die CSS-Datei stammt aus dem Docbook-Entwicklungspaket der e-novative GmbH (siehe Linksammlung)

## Profiling

- Hintergrund: Dokumentationsvarianten
  - Varianten einer Dokumentation redundanzarm und konsistent pflegen
- DocBook-Antwort:
  - Inhaltsbezogenes Kennzeichnen relevanter Elemente mit geeignet definierten Standard-Attributen
  - Ausblenden der nicht gewünschten Elemente bei der Verarbeitung!
- DocBook-Attribute für *profiling*:
  - OS, Arch, Vendor
    - Nur für das Betriebssystem / die Systemarchitektur / den Hersteller relevant, das/die/der im Attributwert steht
  - Userlevel
    - Nur für eine bestimmte Zielgruppe gedacht (z.B. Anfänger, Experten, ...)
  - Lang
    - Das Sprach-Attribut kann auch der selektiven Darstellung dienen
  - Condition: Das generische Profiling-Attribut



## Profiling

- Aufgabe
  - Kennzeichnen Sie drei Absätze Ihrer Wahl mit dem Attribut "userlevel".
  - Wählen Sie Passagen für Experten, ohne die der Gesamttext aber noch lesbar bleibt.
  - Erzeugen Sie nun HTML-Output, in dem diese Passagen ausgeschaltet sind.
    - Suchen und verwenden Sie dazu die geeigneten Stylesheets
    - Setzen Sie ggf. geeignete Parameter!
  - Machen Sie dasselbe für PDF-Output. Der Weg sollte identisch sein!
- Bemerkung:
  - Die Tests des Kursleiters verliefen hier negativ. Sollte es Ihnen hier ebenso ergehen, dann brechen Sie diesen Teil ab.
  - Wer das gewünschte Verhalten erzielt: Bitte den Weg mitteilen! :-)

# Schablonenregeln überschreiben

- Abschlussaufgabe (nur HTML-Kontext)
  - Situation:
    - Bei der HTML-Ausgabe werden die Literaturhinweise mit [1] etc. in den Text geschrieben (Element "citation"), aber es entstehen keine Links zum entsprechenden Eintrag in der Literaturliste.
  - Wünschenswert:
    - Ein HTML-Anker "a" am jeweiligen Eintrag in der Literaturliste
    - Ein automatisch generierter Verweis bei der Verwendung von "citation"
- Lösungsweg-Skizze
  - Anker setzen
    - Wenn Sie das Standardattribut "id" belegen, wird ein entsprechender HTML-Anker dieses Namens gesetzt.
    - Konvention: Sei *x* das verwendete Label, dann laute id="bibid*x*"
  - Link erzeugen
    - Kopieren Sie die Schablonenregel zu "citation" in Ihr custom stylesheet
    - Erweitern Sie diese um ein Link (Element "a")
    - Bei "chunking" sei href := "bi01.html#bibid*x*", ohne "chunking" reicht "#bibid*x*"
    - Tipp: Verwenden Sie für *x* eine lokale Variable.





# Erläuterungen zur Abgabe

# Abgabe

- Generelle Regeln
  - Sie geben nur "gewünschte" Dateien ab.
    - Die Liste der z.Z. "gewünschten" Dateien finden Sie in Datei "abgaben" oberhalb Ihres Abgabeordners
  - Abgabe = Kopieren erwünschter Dateien in "Ihren" Abgabeordner
    - Kopieren - nicht Verschieben! Abgegebene Dateien werden nach kurzer Zeit aus Ihrem Abgabeordner entfernt.
    - Achten Sie auf die Zugriffsrechte Ihrer Dateien. Abgaben, die der Dozent nicht lesen kann/darf, gelten als nicht abgegeben!
  - Nur in den eigenen Abgabeordner schreiben!
    - Es ist zwar technisch möglich, auch in fremde Abgabeordner zu schreiben, aber dies sollten Sie unbedingt vermeiden.
    - Der "file owner" der Datei muss zur Matrikelnummer passen. Kopieren Sie daher während der Abgabe nur mit Ihrem eigenen "account" und in den richtigen Ordner.



# Abgabe

- Generelle Regeln
  - Ihr Abgabeordner (Verzeichnis)
    - folgt aus Ihrer Matrikelnummer, ihrer Praktikumsgruppe und dem Kürzel des Kurses
    - befindet sich im Ordner "abgaben", Unterordner = Ihre Praktikums-Gruppe. Gibt es nur eine Gruppe, lautet diese stets "a".
  - Beispiel:
    - Ihre MatrNr. sei "123456", das Kurs-Kürzel sei "wba", und es gibt nur eine Gruppe. Dann lautet Ihr Abgabeordner:

      /local0/werntges/kurse/wba/abgaben/a/123456

  - Mehrfachabgabe
    - Sollten Sie nach der Abgabe einer Datei (aber vor Ablauf der Abgabefrist) eine korrigierte Version nachreichen wollen: Einfach nochmal abgeben.
    - Die jeweils letzte Datei überschreibt alle Vorgänger gleichen Namens, frühere Versionen werden nicht gespeichert.

# Abgabe

- Generelle Regeln
  - Ablauf nach der Abgabe
    - Ein Skript sammelt regelmäßig alle abgegebenen Dateien ein, sofern sie in der Datei "abzugeben" aufgeführt sind.
    - Diese werden in einen Ihnen nicht zugänglichen Bereich verlagert und dort vom Dozenten ausgewertet.
  - Kontrollen
    - Nur Dateien, die aus Ihrem Abgabeordner verschwinden, kommen beim Dozenten an und zählen als abgegeben!
    - Eine Log-Datei (z.B. "wba.log") im Gruppenorder führt Protokoll über die Abgaben. Sie können hier leicht erkennen, wann welche Ihrer Dateien übernommen wurden, auch bei Mehrfachabgaben: Ein "grep <MatrNr> <filename>" liefert Ihnen alle Einträge zur gegebenen Matr.Nr.
    - Beispiel: grep 123456 wba.log



# Abgabe

- Generelle Regeln
  - Vor der Abgabe für jeden Datei <myfile> empfohlen, die Sie abgeben wollen:
    - Abgabedatei der Gruppe "profs" zuordnen:
      chgrp profs <myfile>
    - Zugriffsrechte entsprechend setzen (normale Dateien):
      chmod 640 <myfile>

      Zugriffsrechte entsprechend setzen (ausführbare Dateien):
      chmod 750 <myfile>
  - Wirkung:
    - Sie entziehen der Allgemeinheit (insb. Ihren Kommilitonen) alle Zugriffsrechte, gestatten aber den Mitgliedern der Gruppe "profs" (und damit Ihrem Dozenten) den Lesezugriff. Sie selbst behalten alle Rechte.
  - VORSICHT:
    - "chgrp" muss funktionieren, ansonsten entziehen Sie auch dem Dozenten die Leserechte! Bei Probleme mit "chgrp" bitte einen Sysadmin fragen.

## Abgabe

- Für diesen Meilenstein abzugeben:
  - A) Quelldateien
    - unterbau.xml: Immer
    - abspeck.ent, ainfo.ent, a_u_e.ent, biblio.ent, einer.ent, init.ent, innen.ent, profiliert.ent, wieder.ent: Falls Sie Entities verwendet haben
  - B) Makefile
    - Makefile: Immer; bitte nur ein Makefile! Das Makefile sollte sowohl HTML- als auch PDF-Output erzeugen können und auch die Benutzung einiger XSL-Optionen enthalten.
  - C) Custom-Stylesheet(s)
    - custom.xsl: Entweder dieses, oder
    - custom1.xsl, custom2.xsl: jene beiden (z.B. wenn Sie für HTML und FO/PDF verschiedene anlegten)
- Bei Problemen
  - Bitte E-Mail an mich (vor Abgabeschluss!)



## Abgabe

- **Abgabeschluss** für diesen Meilenstein:
  - Wie mündlich vereinbart:

**Donnerstag, der 15. 4. 04, < 12:00 Uhr**