



# 7363 - Web-basierte Anwendungen: **Meilenstein 1**

Umgang mit DocBook  
am Beispiel eines c't-Artikels



# Etappe 1

---



- Die Anweisungen für Etappe 1 erfolgen auf drei Weisen
  - Mündlich und auf Folien während des Vorlesungsteils
  - Mündlich während des Praktikums
  - Kompakt auf den folgenden Folien
  
- Zur Arbeitsweise
  - Selbständiges Erschließen des Stoffs ist erforderlich (und dem Niveau eines Vertiefungskurses angemessen)
  - Keine „Kochrezepte“ gegeben, sondern
    - Ausgangspunkte
    - Ziele
    - Hinweise
    - Informationsquellen
  - Arbeitsumfang
    - Ist ausgelegt auf Zweier-Teams



# Etappe 1

---



- Vorbereitungen
  - Vorlagen (unterbau.xml, \*.ent, Makefile\*, \*.css) aus `~werntges/lv/wba/01` in eigenes Arbeitsverzeichnis kopieren
  - Ebenso die Quelldaten des c't-Artikels (HTML-Quellcode + Bilder klein/groß) kopieren
  - Entscheidung: Quelldaten mit Entities oder „monolithisch“ erzeugen?
  - Entscheidung: `sect1/sect2/sect3` oder `section/section/section`?
- Vorarbeiten
  - Markup aus HTML-Vorlage entfernen (!)
  - Abschnitt „Im nächsten Heft“ kann entfallen – entfernen!
  - Bestandteile umsortieren (Anhänge in Text einbetten, evtl. Zerlegung)
- DocBook-Markup einfügen
  - Der Hauptteil; Nachdenken und ggf. Testen erforderlich!
  - Arbeitsteilig vorgehen: Jede(r) bearbeitet ca. die Hälfte!
  - Grundlagen: DocBook – The Definitive Guide, TLDP, KDE Author guides
  - Strukturieren Sie konsequent nach abstrakten Inhalten, nicht nach Aussehen!
  - Checkliste beachten (s.u.)



# 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 Beispieltexes?!
- Literaturverzeichnis (2 Einträge)?



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



- 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...
  - FOP 0.92beta
    - Das Makefile lässt sich leicht auf die neue Beta-Version des FOP umstellen



- 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) Silbentrennung?
  - 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.
  - Prüfen Sie sowohl das Ergebnis von FOP 0.20.5 als auch von 0.92beta! Welches überzeugt mehr?





# HTML-Ausgabe mit "chunking"

---



- Tauschen Sie im Makefile "docbook.xsl" gegen "chunk.xsl"
- Erzeugen Sie erneut den HTML-Output
- Wirkung?
  
- Bemerkung: Fortsetzung zu „chunk“ folgt...



# Etappe 2: *Customizing* des Outputs

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



- Schritt 1
  - Bisher konzentrierten wir uns auf die Frage der korrekten Erfassung unserer Texte als DocBook-Dokument.
  - Die Erzeugung von Ausgabe diente primär der Kontrolle der Erfassung.
  - Grundlage bildete Norman Walsh's Buch „DocBook: The Definitive Guide“
- Schritt 2
  - Tatsächlich spielt die Erzeugung der verschiedenen möglichen Ausgaben eine große Rolle. Sie ist ein so umfangreiches eigenes Thema, dass sie ein eigenes Buch füllt: „DocBook XSL – The Complete Guide“
  - Wir werden hier ausgewählte Themen exemplarisch behandeln, um die Vielfalt an Gestaltungsmöglichkeiten anzudeuten und um einige dazu erforderliche Techniken kennen zu lernen.
  - Für eingehende Beschäftigung mit diesem Thema ist die Lektüre des o.g. Buchs empfohlen – siehe auch Linksammlung!



- 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, etwa:

```
<para role="meinSonderfall"> ... </para>
```
    - Gestaltung dann per CSS: `.meinSonderfall {font-weight: 600}`
  - Sonderfall "emphasis"
    - Wie werden Texte in "emphasis" normalerweise in HTML dargestellt?
    - Verwenden Sie "emphasis" in Ihrem 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?
  - Analyse des HTML-Codes: Wie wird i.a. „class“ befüllt?



- 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 aus „param.xml“ eine lesbare HTML-Dokumentation.
  - Einfacher: Verwenden Sie die Dokumentation in „DocBook XSL...“ !



- Parameter setzen
  - per Kommandozeile:
    - XSLT-Prozessoren besitzen Optionen, die es gestatten, Parameter und deren Werte an das Stylesheet "durchzureichen".
    - Beispiel xalan: Option `-param name expression`
    - Beispiel xsltproc: Option `--stringparam param_name param_wert`
    - Nur in einfachen Fällen (wenige Parameter) praktikabel
  - per Custom-Script (s.u.):
    - 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.xml`)!



## 1. FO-Ausgabe optimieren

- Stellen Sie das Papierformat ein auf "A4"
- Experimentieren Sie mit 2-spaltigem Textsatz! Ergebnis?

## 2. HTML-Ausgabe variieren

- Arbeiten Sie mit "chunking"
- Aktivieren Sie die Navigations-Icons
  - Tipp: Erzeugen 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?
- Variante: Färben Sie alle Dateinamen blau!



- 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">unterbau.css</xsl:param>

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

  <xsl:template name="user.footer.content">
    <div id="customfooter">
      Copyright &#169; 2004, 2006 Fachhochschule Wiesbaden.
      All rights reserved.
    </div>
  </xsl:template>
</xsl:stylesheet>
```



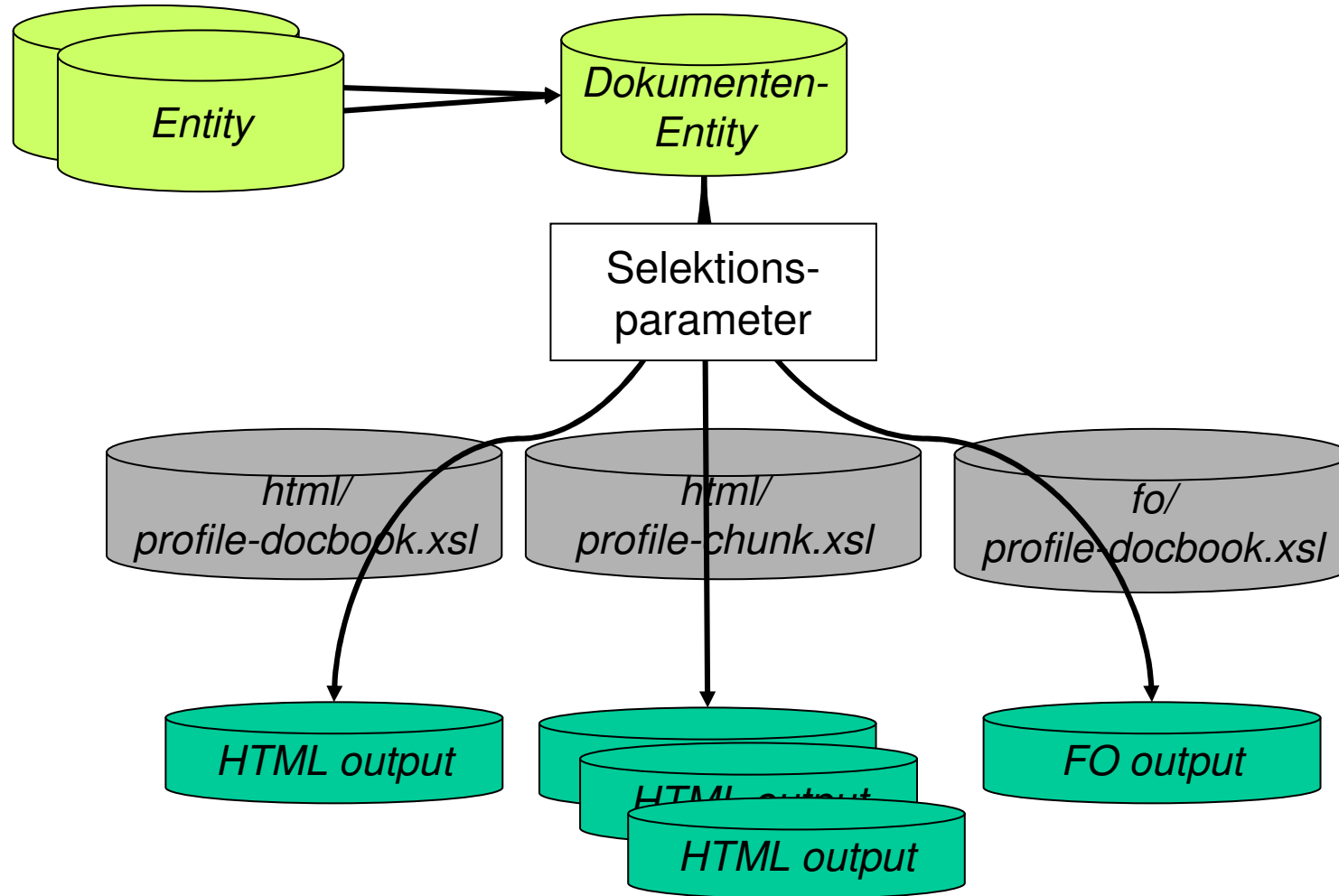
- Wirkung
  - HTML-Erzeugung, "chunking"-Variante
  - Definition einer CSS-Datei für die HTML-Ausgabe
  - Definition eigener Header- und Footer-Texte/Ausgaben
- Anwendung
  - Testen Sie die Wirkung dieses Stylesheets
  - Testen Sie ebenfalls die Wirkung der CSS-Datei
- Quellen
  - a) Ihre Datei „unterbau.css“ mit 1...2 CSS-Anweisungen
  - b) Die CSS-Datei „e-novative.css“ aus dem Docbook-Entwicklungspaket der e-novative GmbH (siehe Linksammlung)
- Hinweise
  - Testen Sie primär mit (b). Was stört (noch)?



- Hintergrund: Dokumentationsvarianten / optionale Texte
  - 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

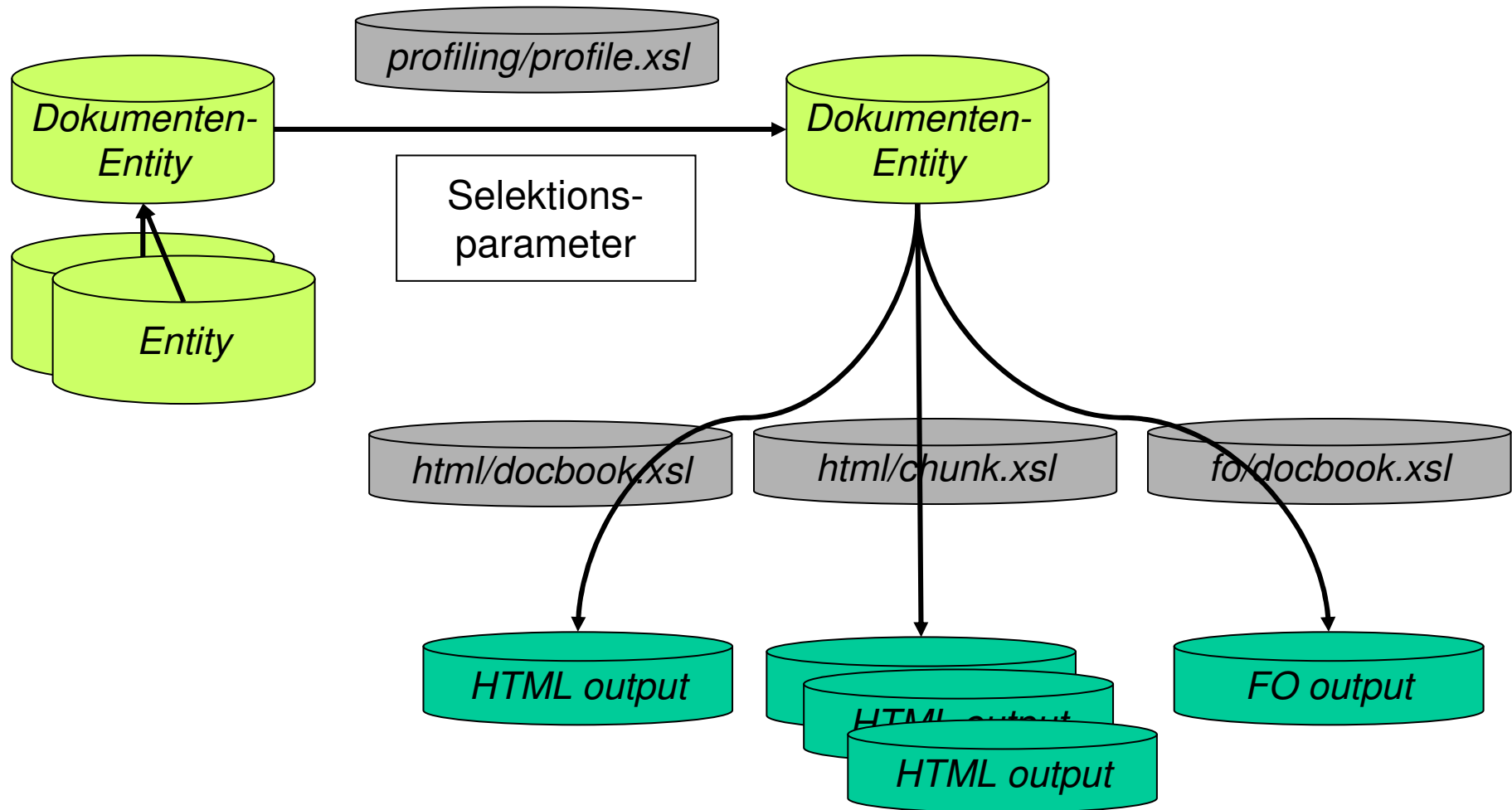


# Profiling: 1-pass processing





# Profiling: 2-pass processing





- 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!
- Bemerkungen:
  - Anleitungen finden Sie in Kap. 25 von DocBook XSL: The Complete Guide – 3rd ed. (siehe Linksammlung)
  - Verwenden Sie „2-pass processing“
  - Wenn möglich, integrieren Sie die neuen Schritte in Ihr Makefile !



- Abschlussaufgabe (nur im HTML-Kontext, bei PDF nicht sinnvoll)
  - 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 bereits von DocBook 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.



- Allgemeine Abgaberegeln beachten
- Keine Abgabe nach Etappe 1
  - Diese sollte allerdings in der ersten Woche abgeschlossen sein, da Etappe 2 darauf aufbaut!
- Abgabe des Gesamtergebnisses (Details siehe nächste Folie)

**bis Montag, den 30.10.06 (Meilenstein 1)**

- **Abgabe durch jeden Teilnehmer / jede Teilnehmerin separat!**
  - Teamarbeit durch Angabe der Autoren (in Kommentaren) markieren





- 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.xml Entweder dieses, oder
    - custom1.xml, custom2.xml jene beiden (z.B. wenn Sie für HTML und FO/PDF verschiedene anlegten)
    - unterbau.css Falls eigene Leistung per CSS-Datei erbracht wurde
- Bei Problemen
  - Bitte E-Mail an den Kursleiter (vor Abgabeschluss!)