Institut für Informatik der Ludwig-Maximilians-Universität München
Lehrstuhl für Programmier- und Modellierungssprachen

Prof. Dr. François Bry, Prof. Dr. Hans Jürgen Ohlbach

Oberseminar 'Knowledge Representation and Markup Languages'

Building an XML Information System (Projektarbeit)

Tobias Kiesling
kiesling@informatik.uni-muenchen.de

02.07.2001

Gliederung

1. Einführung

   1.1 Überblick über das Projekt
   1.2 Das entwickelte System

2. Eingesetzte Software

   2.1 Überblick
   2.2 Cocoon

3. Systemaufbau

   3.1 Erzeugung eines Dokumentes
   3.2 Daten-Modellierung
   3.3 Abstract Page Representation Language (APRIL)
   3.4 Tag-Library
   3.5 APRIL-Transformation

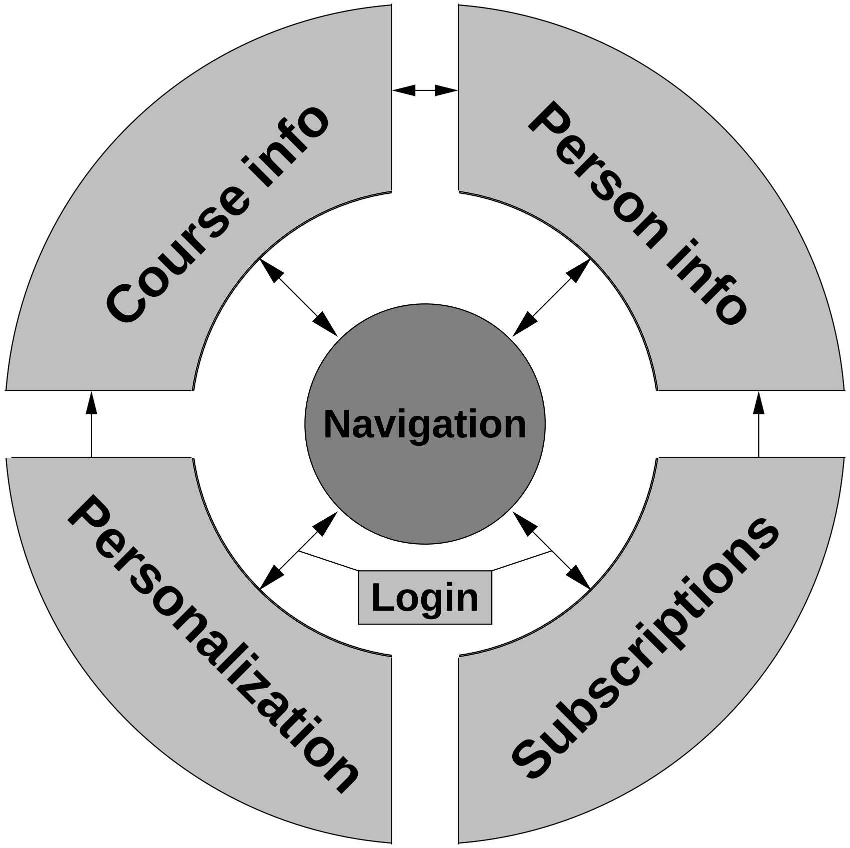
4. Probleme bei der Entwicklung

5. Verbesserungsmöglichkeiten

Überblick

- Informationssystem über Veranstaltungen einer Universität oder eines Instituts
- Entwicklung im Rahmen des Praktikums 'XML und E-Commerce' im Wintersemester 2000/2001
- Entwicklung durch insgesamt 14 Studenten, aufgeteilt in vier Teams:
  - Daten-Modellierungs-Team
  - Content-Management-Team
  - 2 Präsentations-Teams (HTML + WML)
- Entwicklungszeit: 4 Monate
- Ziele des Praktikums:
  - Entwicklung eines funktionsfähigen Prototyps
  - Benutzung von Techniken im XML-Bereich
  - Kennenlernen der Vor- und Nachteile von XML
  - Sammeln von Erfahrungen im Software-Engineering

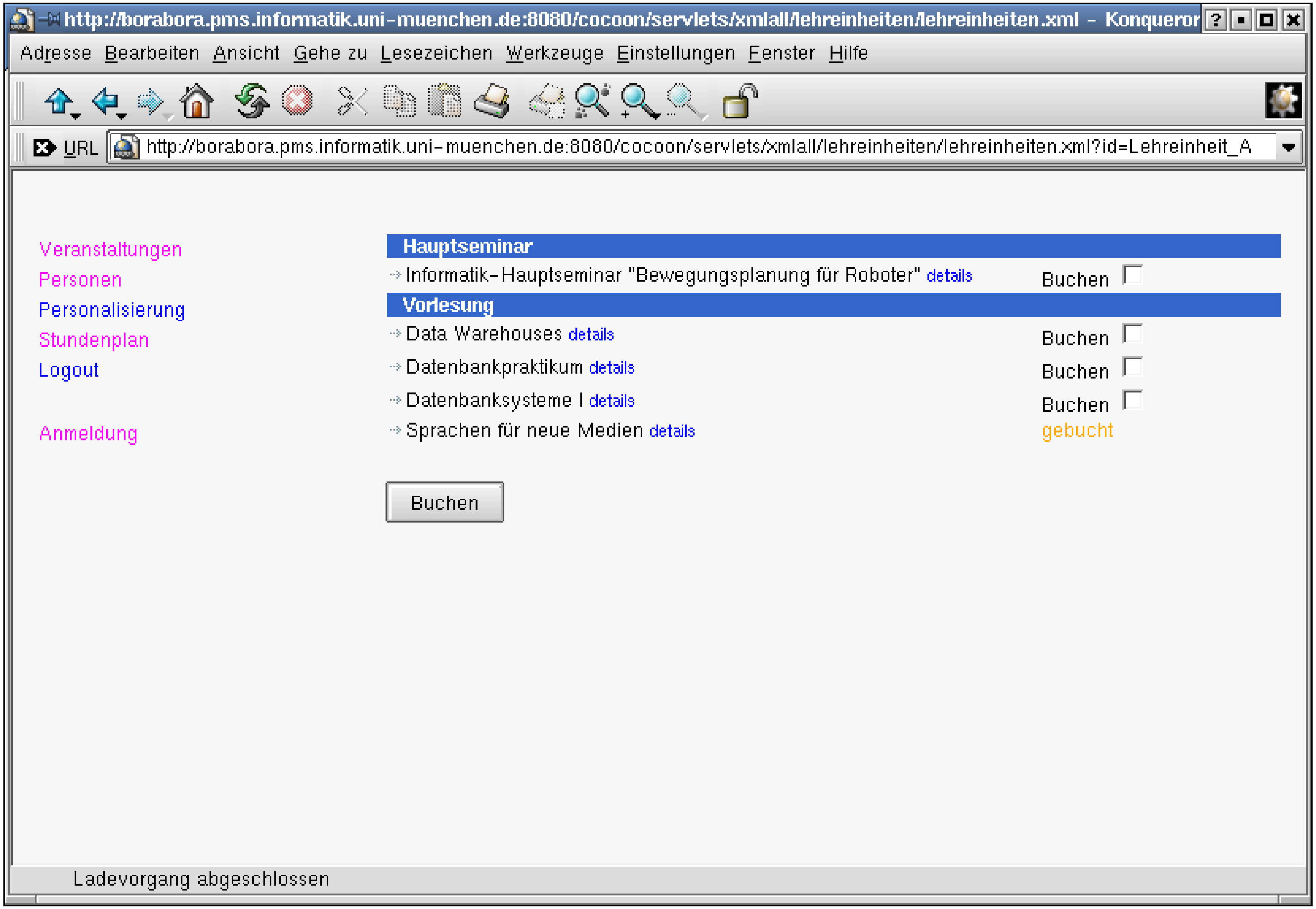
Funktionalität



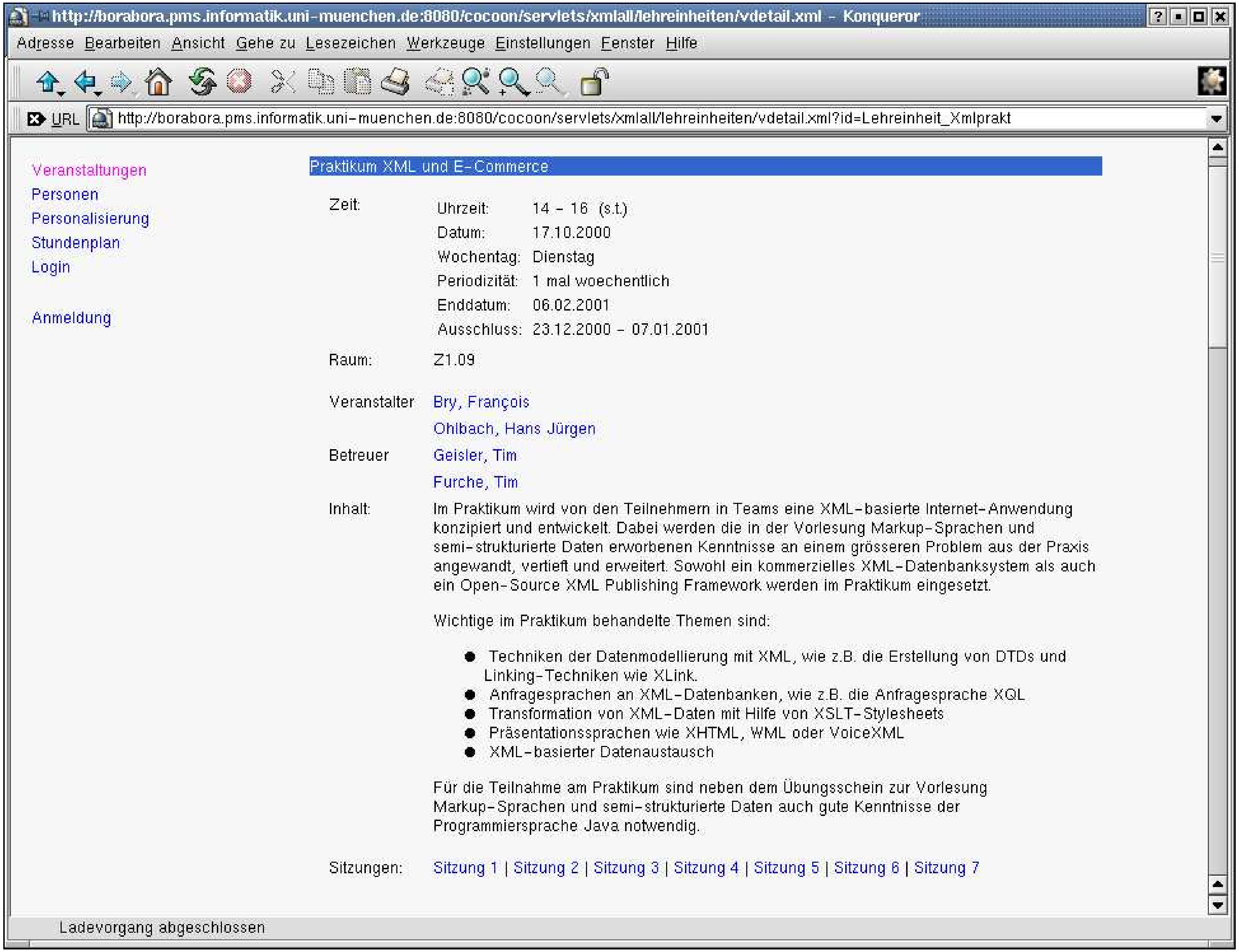
Zentrale Teile des Systems:

- Informationen über Veranstaltungen:
  - Listen
  - Detailseiten
- Informationen über Personen:
  - Liste aller Personen
  - Detailseiten
- Personalisierung:
  - Buchen von Veranstaltungen
  - 'Entbuchen'

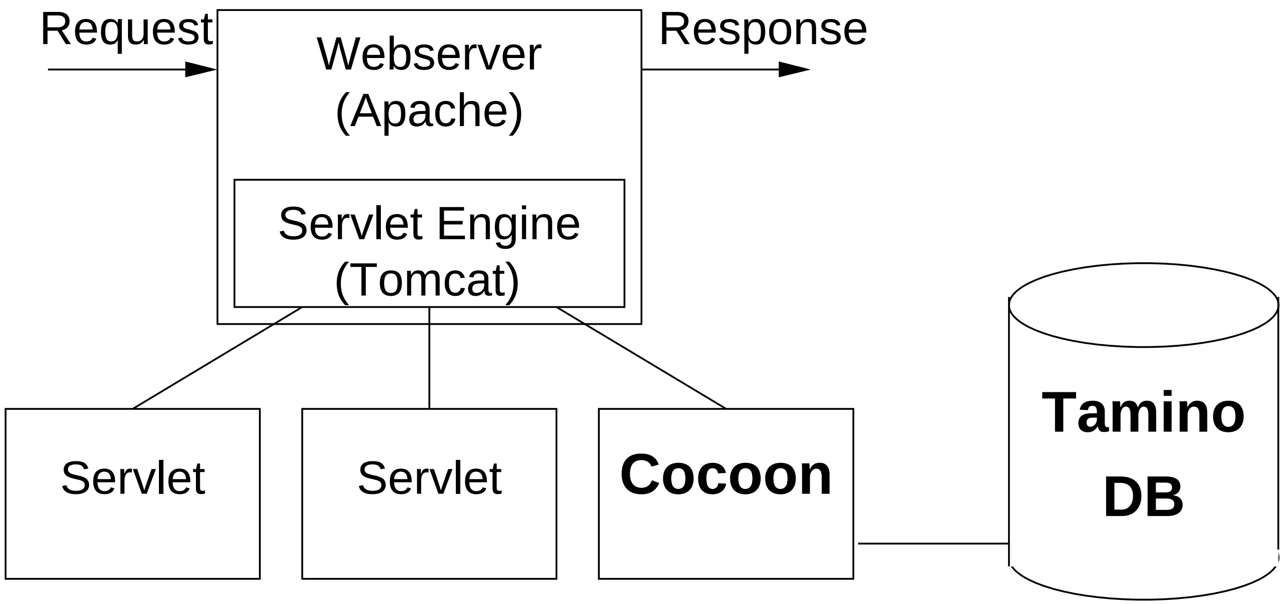
Vorlesungsliste



Detailseite

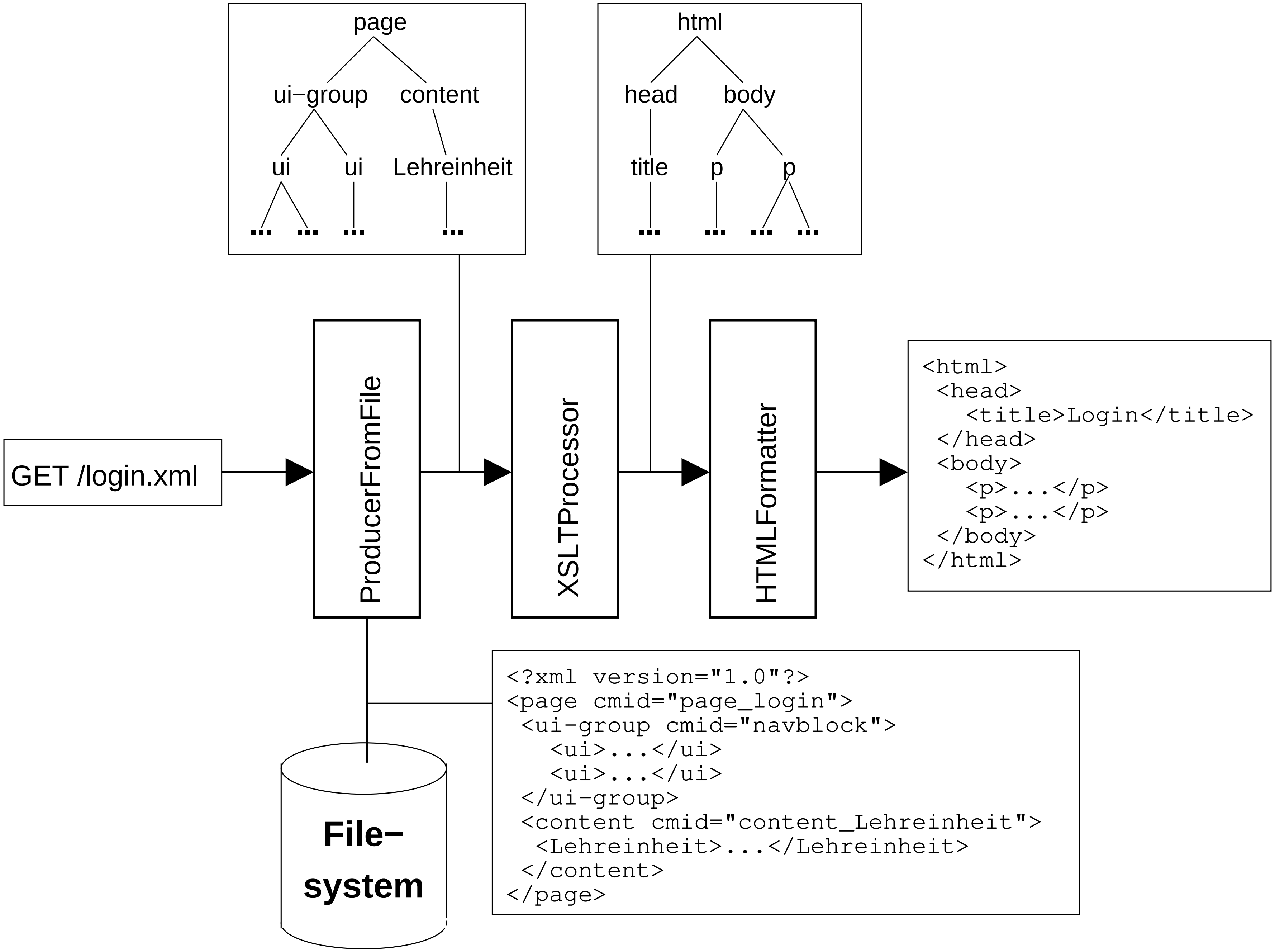


Infrastruktur



- Web-Server Apache 1.3.12
- Servlet-Engine Tomcat 3.1
- Publishing-Framework Cocoon 1.8
- XML-DBMS Tamino 2.2.1.2
- XSLT-Prozessor Xalan 1.2.1

Erzeugung einer Seite



eXtensible Server Pages (XSP)

- Regeln für die Einbettung von Programmiercode in XML-Dokumente
- Vergleichbar zu anderen Server-Pages-Techniken (JSP, ASP)
- Aber: grundsätzlich nicht auf eine einzubettende Sprache festgelegt
- In Cocoon momentan nur Unterstützung für Java

```
<?xml version="1.0"?>
<xsp:page language="java"
     xmlns:xsp="http://www.apache.org/1999/XSP/Core">
 <html>
  <head><title>Time of Day</title></head>
  <body>
   <h3>Time of Day</h3>
   <p>To the best of my knowledge, it's now
      <xsp:expr>new Date()</xsp:expr>
   </p>
  </body>
 </html>
</xsp:page>
```

Tag Libraries

- Tag Library enthält Elemente, die anstelle von XSP-Anweisungen benutzt werden:

```
<p>
  To the best of my knowledge, it's now
  <example:time-of-day format="yy/MM/dd hh:mm:ss aa"/>
</p>
```

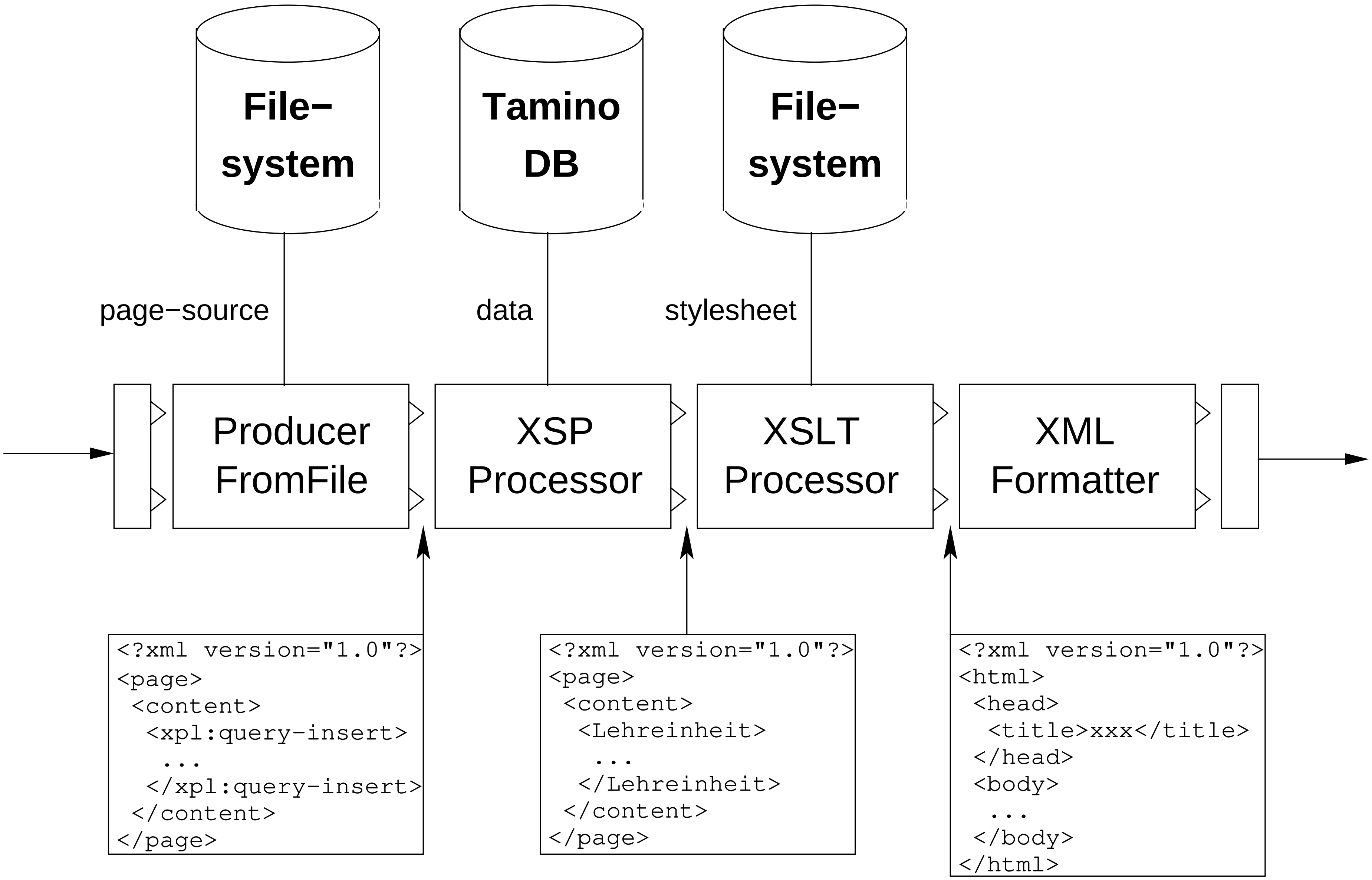
- XSP-Prozessor ersetzt Tags durch Programmcode mit Hilfe von XSLT

- Definitionen von Tag-Libraries sind XSLT-Stylesheets:

```
<xsl:template match="example:time-of-day">
 <xsp:expr>
  formatDate(new Date(), "<xsl:value-of select="@format"/>")
 </xsp:expr>
</xsl:template>
```

⇒ **Trennung von Inhalt und Programmierlogik** ermöglicht

Typische Seite



Datenmodell Beispiel

```
<Lehreinheit ID="Lehreinheit_Xmlprakt">
 <Bezeichnung>Praktikum XML und E-Commerce</Bezeichnung>
 <Zeit Tempore="s.t.">
  <Startuhrzeit>14</Startuhrzeit>
  <Enduhrzeit>16</Enduhrzeit>
  <Datum>17.10.2000</Datum>
  <Regelmaessig>
   <Periodizitaet><woechentlich>1</woechentlich></Periodizitaet>
   <Enddatum>06.02.2001</Enddatum>
  </Regelmaessig>
 </Zeit>
 <Adresse><Raum_Link Raum_ID="Raum_Z1.09"/></Adresse>
 <Beschreibung><p>
  Im Praktikum wird von den Teilnehmern in Teams eine XML-basierte
  Internet-Anwendung konzipiert und entwickelt...
 </p></Beschreibung>
 <Lehreinheit_Link Lehreinheit_ID="Lehreinheit_Xmlprakt_Sitzung01"
                   Rolle="Vorlesung"/>
 <Person_Link Person_ID="Person_FrancoisBry" Rolle="Veranstalter"/>
 <Person_Link Person_ID="Person_TimGeisler" Rolle="Betreuer"/>
 <Lehreinheitstyp>Praktikum</Lehreinheitstyp>
</Lehreinheit>
```

Anforderungen

- Schnittstelle zwischen Content-Management und Präsentation:
  - Entkopplung von Inhalt und Präsentation
  - Präsentation geschieht mit Hilfe von XSLT
  - Deshalb: einheitliches Datenformat notwendig
- Schnittstelle wird gebildet durch XML-Anwendung
- Anforderungen an diese Sprache:
  - Keine Einschränkung der darstellbaren Seiten
  - Unabhängigkeit von einzelnen Präsentationsmedien
  - Unabhängigkeit vom Datenmodell
  - Klare Trennung zwischen Inhalt, Logik und Präsentation
  - Einfache Entwicklung und Wartung von Web-Seiten

$\Rightarrow$ **Abstract Page Representation Language (APRIL)**

Abstract Page Representation Language

- Repräsentation einer Seite durch <**page**>-Element

- Drei verschiedene Typen von Unterelementen:
  - <**content**>:
    - Auf einer Seite darzustellende Informationen
    - Inhalt entspricht dem Datenmodell
  - <**ui-group**>:
    - Dient zur Gruppierung von user-interactions
    - Unterlemente vom Typ <**ui**>:
      - Interaktionsmöglichkeiten eines Benutzers mit dem System
      - 'Abstraktion von HTML Forms'
      - Können 'standalone' oder 'embedded' sein
      - logische Beziehung von 'embedded user-interactions' zu Elementen im Content durch XPath
  - <**meta**>: 'Notanker', kann für verschiedenste Zwecke benutzt werden

- jedes APRIL-Element kann Attribut **cmid** besitzen

  ⇒ Zuweisung von Semantik

Beispiel – HTML


```
<?xml version="1.0" encoding="ISO-8859-1"?>
<html>
 <head><title>Fruit Store</title></head>
 <body>
  <p>
   <a href="/shop_info">About this shop</a>
  </p>
  <p>
   <ul>
    <li><a href="/detail?id=orange_navel">Orange -- Navel</a>
        (Stock: 150 kg)</li>
    <li><a href="/detail?id=apple_boscop">Apple -- Boscop</a>
        (Stock: 300 kg)</li>
   </ul>
  </p>
 </body>
</html>
```

Beispiel – APRIL


```
<?xml version="1.0" encoding="ISO-8859-1"?>
<april:page cmid="fruit_list"
       xmlns:april="http://pms.informatik.uni-muenchen.de/xml/APRIL">
 <april:ui-group cmid="navigation">
  <april:ui cmid="link_shop_info" href="/shop_info"/>
 </april:ui-group>

 <april:ui-group cmid="link_detail" type="embedded">
  <april:ui href="/detail">
   <april:ui-element>
    <april:param-name>id</april:param-name>
    <april:value>@id</april:value>
    <april:location>/april:page/april:content/fruit</april:location>
   </april:ui-element>
  </april:ui>
 </april:ui-group>

 <april:content cmid="content_fruit">
  <fruit id="orange_navel">
   <type>Orange</type><sort>Navel</sort>
   <origin>Spain</origin><stock unit="kg">150</stock>
  </fruit>
  <fruit id="apple_boscop">
   <type>Apple</type><sort>Boscop</sort>
   <origin>Germany</origin><stock unit="kg">300</stock>
  </fruit>
 </april:content>
</april:page>
```

XPL Tag Library

- Auslagerung von benötigter Java-Logik in Tag-Library
- Schnittstelle zwischen Cocoon und Tamino
- Beispiel:

```
<xpl:connection>
 <xpl:url>http://example.com/tamino/xmlprakt</xpl:url>
 <april:content cmid="content_Lehreinheit">
  <xpl:query-insert>
   <xpl:xql-part>Lehreinheit[@ID='</xpl:xql-part>
   <xpl:xql-part type="dynamic">
    <request:get-parameter name="id"/>
   </xpl:xql-part>
   <xpl:xql-part>']</xpl:xql-part>
  </xpl:query-insert>
 </april:content>
</xpl:connection>
```

APRIL-Seite


```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<april:page cmid="fruit_list"
  xmlns:april="http://pms.informatik.uni-muenchen.de/xml/APRIL">
 <april:ui-group type="embedded" cmid="link_buy">
  <april:ui href="/buy">
   <april:ui-element>
    <april:location>//april:content/fruit[stock>0]</april:location>
    <april:param-name>id</april:param-name>
    <april:value type="XPath">@id</april:value>
   </april:ui-element>
  </april:ui>
 </april:ui-group>
 <april:content cmid="content_fruit">
  <fruit id="orange_navel">
   <type>Orange</type><sort>Navel</sort>
   <origin>Spain</origin><stock unit="kg">0</stock>
  </fruit>
  <fruit id="apple_gd">
   <type>Apple</type><sort>Golden Delicious</sort>
   <origin>Germany</origin><stock unit="kg">200</stock>
  </fruit>
 </april:content>
</april:page>
```

Zielseite


```
<?xml version="1.0" encoding="ISO-8859-1"?>

<html>
 <head>
  <title>Fruit List</title>
 </head>
 <body>
  <h3>Fruit List</h3>
  Orange -- Navel<br/>
  <a href="/buy?id=apple_gd">Apple -- Golden Delicious</a><br/>
 </body>
</html>
```

Erster Ansatz

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
     xmlns:april="http://pms.informatik.uni-muenchen.de/xml/APRIL">
 <xsl:template match="page">
  <html>
   <head><title>Fruit List</title></head>
   <body>
    <h3>Fruit List</h3>
    <xsl:apply-templates select="april:content/fruit"/>
   </body>
  </html>
 </xsl:template>
 <xsl:template match="fruit[stock>0]">
  <a href="/buy?id={@id}">
   <xsl:value-of select="type"/> -- <xsl:value-of select="sort"/>
  </a><br/>
 </xsl:template>
 <xsl:template match="fruit">
  <xsl:value-of select="type"/> -- <xsl:value-of select="sort"/>
  <br/>
 </xsl:template>
</xsl:stylesheet>
```

Zweiter Ansatz (1)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:axsl="http://www.w3.org/1999/XSL/TransformAlias"
   xmlns:april="http://pms.informatik.uni-muenchen.de/xml/APRIL">

 <xsl:namespace-alias stylesheet-prefix="axsl"
                      result-prefix="xsl"/>

 <xsl:template match="page">
  <xsl:processing-instruction name="xml-stylesheet">
   href="#styler" type="text/xsl"
  </xsl:processing-instruction>

  <april:page>
   <xsl:copy-of select="@*|april:content|april:meta"/>
   <axsl:stylesheet id="styler" version="1.0">
```

Zweiter Ansatz (2)


```
    <axsl:template match="april:page">
     <html>
      <head><title>Fruit List</title></head>
      <body>
       <h3>Fruit List</h3>
       <axsl:apply-templates select="april:content/fruit"/>
      </body>
     </html>
    </axsl:template>
    <xsl:apply-templates
         select="april:ui-group[@cmid='link_buy']/april:ui"/>
    <axsl:template match="fruit">
     <axsl:value-of select="type"/> -- <axsl:value-of select="sort"/>
    </axsl:template>
   </axsl:stylesheet>
  </april:page>
 </xsl:template>
 <xsl:template match="april:ui-group[@cmid='link_buy']/april:ui">
  <axsl:template match="{april:ui-element/april:location}">
   <axsl:variable name="url" select="concat(
           '{@href}?{april:ui-element/april:param-name}=',
           {april:ui-element/april:value})"/>
   <a href="{'{$url}'}">
    <axsl:value-of select="type"/> -- <axsl:value-of select="sort"/>
   </a><br/>
  </axsl:template>
 </xsl:template>
</xsl:stylesheet>
```

Probleme

- Mächtigkeit der Tamino Anfragesprache X-Query

  ⇒ Vortrag 'Tamino Experience Report' nächste Woche

- Transformation der APRIL-Seiten
  - Schwierig zu machen mit XSLT
  - Einzige gefundene Lösung komplex und schlecht wartbar

- Performanz des Systems (siehe nächste Folie)

- Entwicklung mit Cocoon
  - Teilweise schlechte Dokumentation
  - Wenig bis keine Unterstützung von Debugging

- Session-Handling
  - Vollständig abgestützt auf Cookies
  - Keine Sessions möglich bei 'Cookie-losen' Geräten

Performance

- Hohe Antwortzeiten selbst bei geringer Belastung
- Aufschlüsselung der Gesamt-Verarbeitungszeit in einzelne Schritte:

Schritt	lehreinheiten.xml	vdetail.xml	personen.xml	pdetail.xml
Whole	328	861	970	570
Producer	17 (5.2%)	20 (2.3%)	10 (1.0%)	10 (1.8%)
XSP	110 (33.5%)	460 (53.4%)	587 (60.5%)	319 (56.0%)
XSLT 1	159 (48.5%)	177 (20.6%)	164 (16.9%)	166 (29.1%)
XSLT 2	39 (11.9%)	183 (21.3%)	196 (20.2%)	73 (12.8%)
Formatter	3 (0.9%)	21 (2.4%)	13 (1.3%)	2 (0.4%)

$\Rightarrow$ Über 50% der Gesamtantwortzeit durch DB-Anfrage

$\Rightarrow$ Beide XSLT-Schritte zusammen ca. 40-50%

Verbesserungsvorschläge

- APRIL-Transformation:
  - Andere Transformationstechnik
  - Änderungen an APRIL
- Seitenspezifikationen:
  - Problem: welche Teile einer APRIL-Seite sind fest, welche können sich im Lauf der Zeit verändern?
  - Lösungsmöglichkeit: Spezifikation von Seiten
- Verbessertes Session-Handling (z.B. alternativ URL-Rewriting)
- Performance-Verbesserung:
  - Effizientere APRIL-Transformation
  - Effizientere Datenbankanbindung
- High-Level Tag-Library:
  - Vorhandene Tag-Library $\Rightarrow$ 'Schnellschuß'
  - Spezifisch auf Anwendung zugeschnittene Tag-Library