1B
Übung WME
XML-Verarbeitung und XSLT

Martin Lehmann - Martin.Lehmann4@mailbox.tu-dresden.de

Technische Universität Dresden
Fakultät Informatik
Institut für Software- und Multimediatechnik
Lehrstuhl für Multimediatechnik

Gliederung

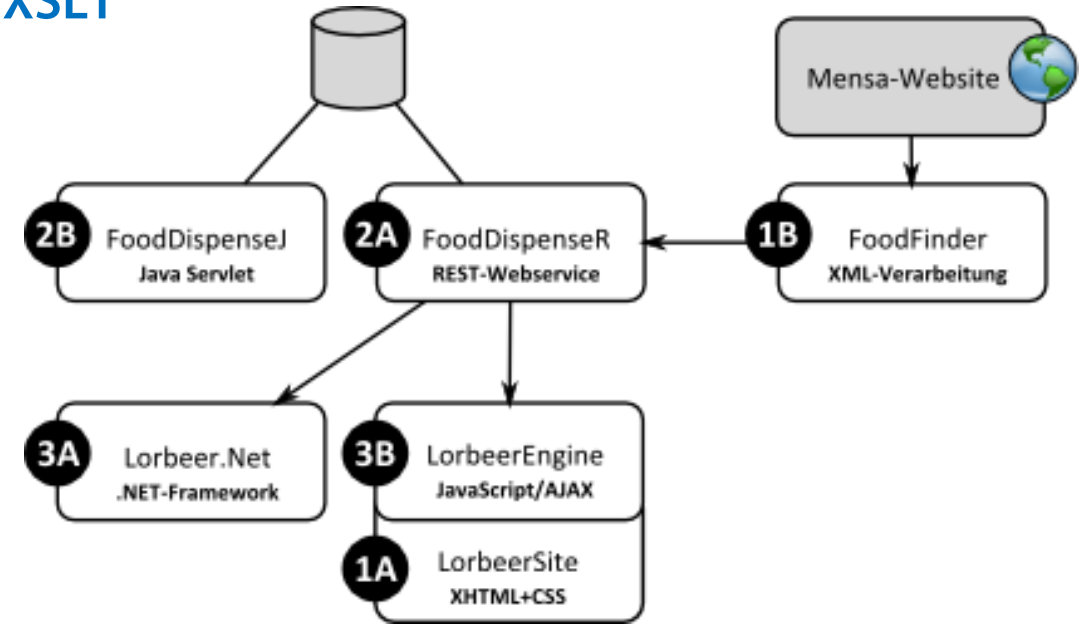
- Organisatorisches
  - Ablauf der Übung
  - Themen
  - Abgabemodalitäten
  - Gruppenzuordnung
- Aufgabenstellung
  - Hinweise zur Umsetzung
- Einstieg ins Thema
  - XML
  - XPATH
  - XSLT
- Links

Organisatorisches: Ablauf der Übung

- Terminplan
  - 19.10.11 Übungseinführung 1B (Pflicht)
  - 26.10.11 Konsultation (Teilnahme freiwillig)
  - 02.11.11 Konsultation (Teilnahme freiwillig)
  - 09.11.11 Konsultation (Teilnahme freiwillig)
  - 14.11.11 Abgabe der Ergebnisse (bis spätestens 13:00 Uhr)
  - …
  - 01.02.12 Abschlussveranstaltung und Präsentation (Pflicht)

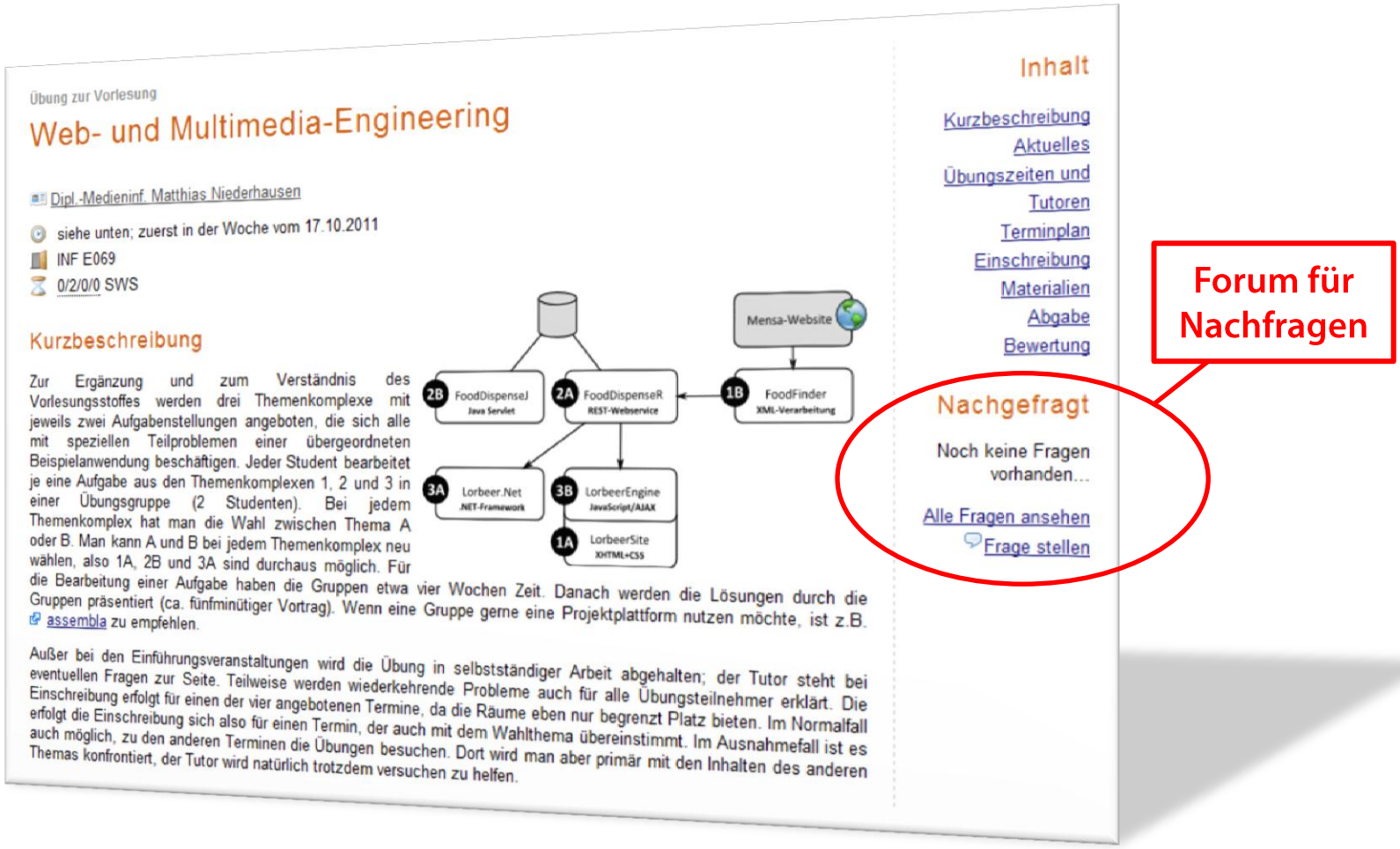
- Details siehe Übungskalender

- Themenkomplex 1
  - 1A) XHTML und CSS
  - 1B) XML-Verarbeitung und XSLT
- Themenkomplex 2
  - 2A) RESTful Webservice
  - 2B) Java Servlets
- Themenkomplex 3
  - 3A) .NET/Silverlight
  - 3B) JavaScript und AJAX



- **Informationen zu den Übungen**

  http://www.mmt.inf.tu-dresden.de/Lehre/Wintersemester_11_12/WME/Uebung/index.xhtml



Organisatorisches: Abgabemodalitäten

- via Upload auf sFTP Server:
  - Servername: serv9.inf.tu-dresden.de
  - Port 22
  - Verzeichnis: /zbv/WME/
  - Authentifizierung mit ZIH-Login

- Abgabe der Lösung
  - Verzeichnis anlegen
    - Ergebnisse_1_B/<Gruppe>_<Matrikel1>_<Matrikel2>
  - Dateien hochladen
    - keine Umlaute in Dateinamen!
  - Zugriffsrechte nicht verändern!

Organisatorisches: Abgabemodalitäten

- Lösung der Aufgabe
  - zwei Studenten bilden ein Team
    - auf der Einschreibliste eintragen!
  - Durchschleifen wird nicht toleriert
  - Abgabetermin ist verbindlich (14.11.2011 - bis 13Uhr)
    - Abgabeverzeichnisse werden nach Ablauf der Bearbeitungszeit gesperrt
  - jede Gruppe hat eine eigenständige Lösung abzugeben
  - ausgewählte Ergebnisse werden am letzten Übungstermin (01.02.2012) von den Gruppen präsentiert

- **Aufgabe 1B:** ***Automatische Datenerfassung vom Online-Speiseplan des Studentenwerks***
  - Das Portal Lorbeerblatt soll Studenten eine Plattform bieten, auf der sie sich über das aktuelle Mensaessen austauschen können. Die Benutzer sollen Kommentare, Bewertungen und eigene Bilder der verschiedenen Speisen einstellen können. Dazu muss zunächst der aktuelle Speiseplan von der Website des Studentenwerks eingelesen werden.

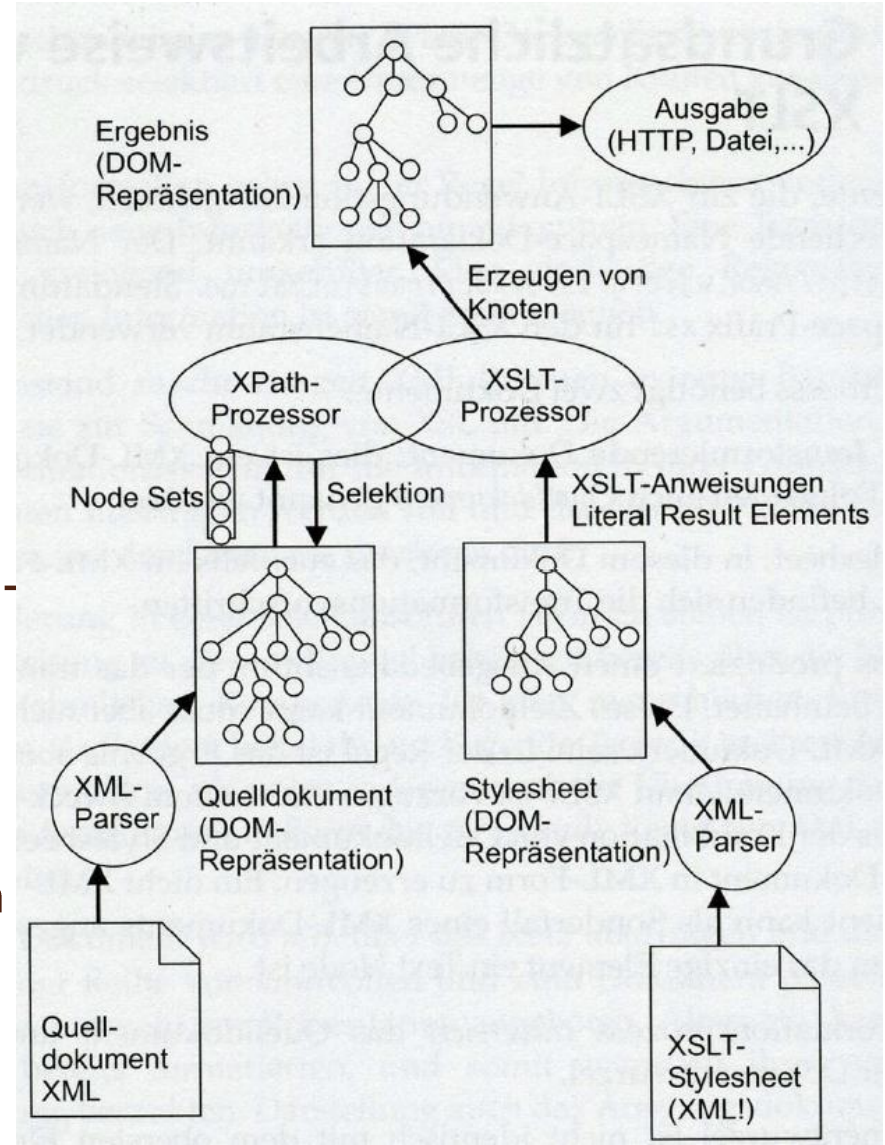
Aufgabenstellung im Detail

- Entwicklung des Content Scrapers für Lorbeerblatt
  - Java-Programm zur Automatisierung des Einlesevorgangs
  - Funktionen:
    - Verbindung mit dem Webserver des Studentenwerks
    - Einlesen der XHTML Informationen von der offiziellen Mensa-Website (http://www.studentenwerk-dresden.de/mensen/speiseplan/)
    - Eindeutiges Identifizieren von Mensen und Wochentagen
    - Strukturiertes Auslesen der Tagesspeisen
    - Erzeugung und Speicherung der strukturierten Informationen in Java-Objekte
  - Der beste Entwurf wird im weiteren Verlauf der Übung weiterverwendet!

Aufgabenstellung im Detail

- **Auslesen relevanter Daten der Mensen**
  - **Zu erfassende Daten:**
    - Name der Mensa
    - Name des Gerichts
    - Preise, getrennt für Studenten und für Mitarbeiter
    - Inhaltsstoffe unter „Infos“, als Liste
    - URL des zugehörigen Bildes (falls vorhanden)
    - URL der Detailseite der Speise

- **Arbeitsweise des XSLT-Prozessors im Zusammenspiel mit XPath:**
  - Einlesen des Quelldokuments und des XSLT-Stylesheets
  - XML-Parser erzeugt DOM-Baum von dem Quelldokument und dem Stylesheet
  - XSLT-Prozessor verwendet die XPath-Ausdrücke innerhalb der XSL-Datei zur Selektion der DOM-Knoten (Beginn bei dem Wurzelelement)
  - Für den Bearbeitungsknoten werden Regeln im Stylesheet gesucht und angewandt
  - Ergebnisgenerierung



Aufgabenstellung im Detail

- Erstellung eines XSL-Stylesheets (*.xsl) für die Transformation
  - XSL-Stylesheetdeklaration (Version und Namespaces) im Kopfbereich des XML-Dokuments mit <xsl:stylesheet>
  - Formatdefinition des Ausgabeformats mit <xsl:output>
  - Definition von Transformationsregeln, die auf das Ausgangsdokument angewendet werden mit <xsl:template>
  - Strukturierung relevanter Daten innerhalb der Transformationsregeln
  - Nutzung von XPath-Ausdrücken, um auf die Knotenmenge des Ausgangsdokuments zuzugreifen

Aufgabenstellung im Detail

- JAXP-API:
  - zum Validieren, Parsen, Generieren und Transformieren von XML-Dokumenten
  - DOM-Parser muss wie folgt erzeugt werden:

```
DocumentBuilderFactory fac = DocumentBuilderFactory.newInstance();
fac.setFeature(„http://apache.org/xml/features/nonvalidating/load-external-dtd“, false);
DocumentBuilder db = fac.newDocumentBuilder();
```

  - Parsen der Quelldatei und des XSLT-Stylesheets
  - Anwendung der XSL-Transformation auf die DOM-Bäume

Aufgabenstellung im Detail

- Randbedingungen
  - gültiges XML: einfach zu prüfen mit http://validator.w3.org (bei der Betrachtung der XML-Datei im Browser -> XML-Deklaration & XML-Namensraum werden nicht angezeigt)
  - Java-Objekte müssen reale und korrekte Daten der Mensawebsite beinhalten
    - http://www.studentenwerk-dresden.de/mensen/speiseplan/
  - Lizenzrechte bei verwendeten Fremdinhalten beachten!
- Abgabe
  - XSL-Stylesheet (transform.xsl)
  - gepackte Quellen (z.B. als gepacktes Eclipseprojekt)
  - kurze Ausführungsbeschreibung

Aufgabenstellung im Detail

- Material
  - lorbeerblatt-core.jar
    - enthält vorgefertigte Javaklassen zur Speicherung der geparsten und selektierten Informationen von der Mensawebsite
  - XML-Libraries (Apache xerces2):
    - XML-Parser zum Navigieren durch den verzweigten DOM-Baum, Modifizieren und Generieren von XML-Daten
    - muss mit in das Projekt eingebunden werden!

- Editoren und Entwicklungsumgebungen
  - Notepad++
    - Texteditor mit Syntax-Highlighting
    - klein und schnell
    - zur Erstellung der XSL
  - Eclipse
    - komplexe Entwicklungsumgebung, ursprünglich für Java
    - für die Umsetzung der Aufgabe unabdingbar
    - bietet die Möglichkeit die Anwendung in ein ApplicationServer zu deployen (vorarbeiten für die nächsten Themenkomplexe)

Erstellung einer Applikation in Eclipse

- Dynamic Web Project erstellen
- als Target runtime den Apache Tomcat 7.0 angeben (vorher unter http://tomcat.apache.org/ runterladen)
- unter WebContent/WEB-INF/lib alle benötigten JARs speichern
- unter WebContent/WEB-INF transform.xsl und die web.xml erstellen
- in dem Package src werden alle verwendeten Klassen hinterlegt
- Erstellung eines FoodFinderServlets (erbt von javax.servlet.http.HttpServlet) in einem eigenen Package (Bsp.: de.mmt.lorbeerblatt.foodfinder) und Implementierung der doGet-Methode
- in der web.xml erfolgt das Servlet-Mapping von der URL http://localhost:8080/{Projektname}/FoodFinder auf diese Servletklasse

- **Aufbau einer einfachen web.xml:**

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xmlns="http://java.sun.com/xml/ns/javaee"
         xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
            http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
    <display-name>FoodFinder</display-name>
    <servlet>
        <servlet-name>FoodFinder</servlet-name>
        <servlet-class>de.mmt.lorbeerblatt.foodfinder.FoodFinderServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>FoodFinder</servlet-name>
        <url-pattern>/FoodFinder</url-pattern>
    </servlet-mapping>
</web-app>
```

- **Starten der Applikation:**
  - Projekt -> Run As / Run on Server
  - Tomcat v7.0 Server auswählen
  - Server runtime environment setzen (Pfad zum entpackten Tomcat 7 einstellen)
  - nach erfolgreichem Start ist die Anwendung unter http://localhost:8080/{Projektname}/FoodFinder erreichbar

- XML Geschichte
  - seit 1998 erste Empfehlung vom W3C
  - Auszeichnungssprache für die Darstellung hierarchischer Daten
  - zentrale Elemente Wohlgeformtheit, Elemente, Tags
  - Meta-Sprache mit Meta-Grammatik zur Schaffung beliebiger Sprachen
  - Daten werden hierarchisch gegliedert
  - XML bietet generische Möglichkeiten zur Transformation und Parsen
  - inzwischen haben sich viele XML-Sprachen durchgesetzt:
    - XHTML, SVG, XAML, Atom, SOAP

- **Wohlgeformtheit**
  - Dokumente haben immer ein Wurzelelement
  - Tags markieren einzelne Bereiche in Dokumenten und müssen immer geschlossen werden
  - Tags dürfen verschachtelt, aber nicht verschränkt sein
  - Tags können Attribute haben

```
<?xml version="1.0" encoding="UTF-8"?>
<html>
<head>
<title>Insert title here</title>
</head>
<body>
Hallo <b>schöne</b> neue Welt!
</body>
</html>
```

- XML-Verarbeitung mittels Parsen
  - durch die innere gegebene Struktur von XML gibt es definierte Techniken zur Verarbeitung
  - Zugriff auf XML-Knoten mittels DOM
- direkte XML-Verwendung
  - als Datenaustauschformat (Webservices, Ajax)
  - Als Datenspeicherformat (XML Databases, file based)
  - im Browser mittels integrierter XSLT Transfomation
- siehe auch: http://www.w3schools.com/xml

- XSL Transformation
  - Seit November 1999 Empfehlung des W3C
  - Zweck: Transformation eines XML-Quelldokuments in ein Zieldokument
    - Häufigster Anwendungsfall: Transformation in XHTML
  - Liegt selbst in XML-Syntax vor, d.h. jedes XSLT-*Stylesheet* ist ein XML-Dokument
  - Bei der Transformation können die Informationen des Quelldokumentes z.B. gefiltert, sortiert und nummeriert werden.

- XML Path Language (XPath)
  - Dient der Adressierung beliebiger Knoten in XML-Dokumenten:
    - Elemente
    - Attribute
    - Textknoten
  - Seit November 1999 Empfehlung des W3C
  - Definition von Achsen (*axes*) und Funktionen zur Navigation im XML-Dokument. Axen sind Dimensionen zur Adressierung von Knoten.
  - Mit XPath können sowohl einzelne Knoten (*node*), als auch Knotenmengen (*node-set*) adressiert werden
  - siehe auch: http://www.w3schools.com/xpath

- **XSLT Workflow**
  - **mit XSLT-Anweisung `xsl:template` werden Teile des Ausgangsdokuments gematcht, dabei wird XPath verwendet, um die Ausgangsstruktur zu adressieren**
    - `<xsl:template match=„//table/tr“>`
    - `...`
    - `</xsl:template>`
  - **innerhalb eines Match-Blocks werden dann die Ausgangsstrukturen verarbeitet oder aber transformiert bzw. andere Templates aufgerufen**
  - **Möglichkeiten zum Überprüfen auf Bedingungen mittels `<xsl:if>`**
  - **Konstantenbelegung mittels `<xsl:variable>`**

  - **`siehe auch:` http://www.w3schools.com/xsl**

Zum Weiterlesen…

- Tutorien & Co. (XML, XSLT, XPath)
  - http://www.w3schools.com/
  - http://www.zvon.org/
  - http://www.w3.org/TR/xpath/
  - http://www.w3.org/TR/xslt/
  - http://www.oreilly.de/artikel/java_xslt_tips.html
- kostenloses Repository
  - http://www.github.com
  - http://www.assembla.com
- Kontakt
  - Martin.Lehmann4@mailbox.tu-dresden.de