ZHAW Seminararbeit
XML Datenbanken
- - -
Lösen eines begrenzten Datenhaltungsproblem mit einem nativen XML-DBS am Beispiel von Schuh-Steckbriefen

Michael Ott, i10b

17. Juni 2014

Inhaltsverzeichnis

1. Einleitung

1.1. Aufgabenstellung (ZHAW)

1.1.1. Ausgangslage

Jeder männliche Partner in einer Beziehung kennt das Problem: Die Schuhsammlung der Frau. Aus Sicht der Frau ist grundsätzlich nie ein passender Schuh vorhanden (Farbe falsch, Schnitt unpassend, Absatz zu hoch,...).
Oft stellt sich dann aber nach einem weiteren Schuh-Kauf heraus, dass es eigentlich doch schon einen praktisch identischen Schuh in einer etwas in Vergessenheit geratenen Kiste gegeben hätte.

Um jederzeit einen Überblick über die verschiedenen Schuhmodelle zu haben, soll jeder Schuh anhand eines Steckbriefes (Aufbewahrungsort, Grösse, Farbe, Absatzhöhe, Schnitt, max. Tragedauer,...) erfasst werden.
So lässt sich jederzeit bei Bedarf gezielt prüfen, ob und an welchem Aufbewahrungsort ein bestimmtes Schuhmodell bereits vorhanden ist.

1.1.2. Ziel der Arbeit

Es soll untersucht und geprüft werden, ob eine native XML-Datenbank die Anforderungen an die Erfassung und die Suche von Schuh-Steckbrief - Daten erfüllen kann.
Die Erfassung soll nach Pflicht- und optional Angaben eingeteilt werden, auch ist zu prüfen ob und in welcher Form Bilder zum Steckbrief abgelegt werden können.
Um die Datenqualität zu gewährleisten, müssen Eingaben validiert werden können.
Es wird geprüft, ob und in welcher Form das seitens XML-DBS möglich ist.
Bei der Suche soll es möglich sein, gezielt einen einzelnen oder alle Steckbriefe mit einem bestimmten Merkmal zu suchen.

1.1.3. Aufgabenstellung

1. Requirement Engineering
   - Was wird bezüglich XML-DBS benötigt
   - Formulieren der konkreten Anforderungen
2. konzeptioneller Entwurf
   - Wie soll das Datenbankschema aussehen
   - Wie muss die Erfassung und die Suche umgesetzt werden
3. Datenbankdefinition - Datenbank gemäss Konzept erstellen
4. Implementierung
   - Erfassen von Testdaten in der XML-Datenbank
   - Schuh-Erfassung und Suche
5. Zielüberprüfung
   - Schuh-Erfassung und Suche gemäss den definierten Anforderungen

1.1.4. Erwartetes Resultat

1. Requirement Engineering
   - Dokumentierte Anforderungen (XML-DBS resp. Anwendung)
2. konzeptioneller Entwurf
   - Dokumentiertes Datenbankschema
   - Konzept für Ablauf der Erfassung und Suche
3. Datenbankdefinition
   - Datenbankschema erstellt
   - XML-Datenbank funktionsfähig erstellt
4. Implementierung
   - Schuh-Erfassung und Suche sind umgesetzt
5. Zielüberprüfung
   - Schuh-Erfassung und Suche nach gezielt wählbaren Kriterien funktionieren gemäss den definierten Anforderungen

Zusätzlich zur Umsetzung sind gemäss Aufgabenstellung ZHAW sämtliche Anforderungen und Konzepte in einem Seminarbericht zusammengefasst.

1.1.5. Termine

Datum	Schritt
12.03.2014	KickOff-Besprechung
20.04.2014	Abgabe Aufgabenstellung
10.06.2014	Abgabe Seminarbericht
17.06.2014	Präsentation der Arbeit

2. Anforderungen

In diesem Kapitel sind alle Anforderungen an das System aufgeführt.

2.1. Stakeholder

Für die Definition der Anforderungen wird als Stakeholder dank einer umfangreichen Schuhsammlung die Frau des Autors berücksichtigt. Sie wird folgend *Benutzer* genannt.

Die technischen Anforderungen werden durch den Author definiert, resp. sind bereits durch die Aufgabenstellung vorgegeben.

2.2. Use-Case-Diagramm

Das folgende UseCase-Diagramm zeigt die Aktionen welche vom Benutzer ausgelöst werden können

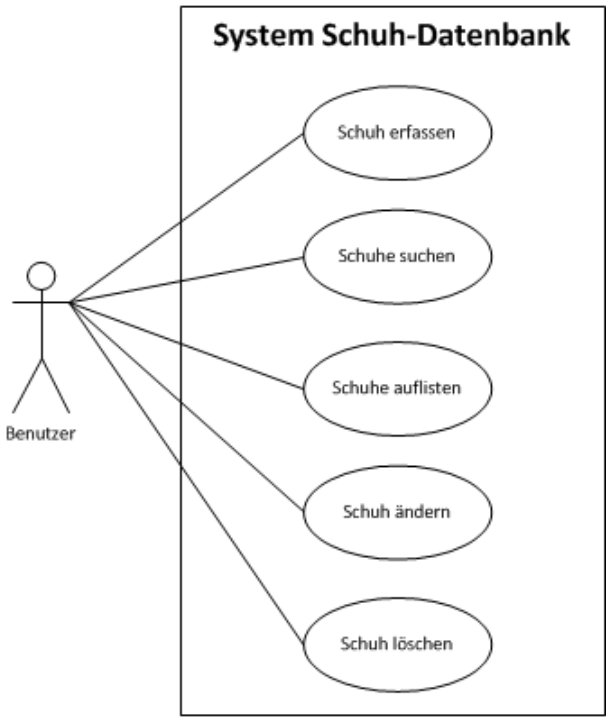

Abbildung 2.1.: Use-Case-Diagramm Benutzer

2.3. Allgemein

Die Anforderungen sind strukturiert nach folgendem Schema aufgelistet:

ID	ID der Anforderung (Req- + 0..999)
Bezeichnung	Titel der Anforderung
Beschreibung	Kurze Beschreibung
Umsetzung	Pflicht, Optional, Informativ
Vorbedingung	ID einer anderen Anforderung
Zusatz-Info	Verweis auf Kapitel in Dokumentation oder sonstige Quelle
Status	Angabe über den Realisierungsgrad

2.4. Nichtfunktionale Anforderungen

Die folgenden Anforderungen sind in der Aufgabenstellung vorgegeben:

ID	Req-0001
Bezeichnung	natives XML-DBS
Beschreibung	Die Datenablage soll ein natives XML-Datenbanksystem eingesetzt werden
Vorbedingung	-
Zusatz-Info	-
Status	Umgesetzt

ID	Req-0002
Bezeichnung	Ablagestruktur
Beschreibung	Jeder Steckbrief wird im XML-Format abgelegt
Vorbedingung	Rep-0001
Zusatz-Info	-
Status	Umgesetzt

ID	Req-0003
Bezeichnung	Zugang mittels Browser
Beschreibung	Der Benutzer soll die Daten via Browser abrufen können, damit auch ein Zugriff mit mobilen Geräten möglich ist
Vorbedingung	Req-0002
Zusatz-Info	-
Status	Umgesetzt

2.5. Funktionale Anforderungen

Für diese Arbeit wird auf ein Backend für die Verwaltung der Datenbank und eine Benutzerverwaltung verzichtet.

Die folgenden Pflicht-Anforderungen entsprechen daher den Anforderungen eines Benutzers.

ID	Req-0004
Bezeichnung	Steckbrief erfassen
Beschreibung	Benutzer kann mittels vorgegebener Maske einen Schuh erfassen.
Vorbedingung	Req-0001, Req-0003
Zusatz-Info	Pflicht-Felder: Marke, Modell, Typ, Farbe, Grösse, Absatzhöhe Optionale Angaben: Lagerort, Notiz
Status	Umgesetzt

ID	Req-0005
Bezeichnung	Steckbrief auflisten
Beschreibung	Sämtliche erfassten Steckbriefe sollen in einer Liste angezeigt werden
Vorbedingung	Req-0004
Zusatz-Info	-
Status	Umgesetzt

ID	Req-0006
Bezeichnung	Steckbrief-Suche
Beschreibung	Es soll nach jedem Attribut des Steckbriefes gesucht werden können. Als Resultat soll eine Liste mit sämtlichen gefundenen Einträgen erscheinen
Vorbedingung	Req-0004
Zusatz-Info	-
Status	Umgesetzt

2.6. Optionale Anforderungen

Die folgenden optionalen Anforderungen sind vom gleichen Stakeholder (Kapitel 2.5) definiert.

ID	Req-0007
Bezeichnung	Steckbrief löschen
Beschreibung	Ein wählbarer Steckbrief soll gelöscht werden können
Vorbedingung	Req-0006
Zusatz-Info	-
Status	Umgesetzt

ID	Req-0008
Bezeichnung	Steckbrief anpassen
Beschreibung	Ein wählbarer Steckbrief kann angepasst werden
Vorbedingung	Req-0006
Zusatz-Info	-
Status	offen

ID	Req-0009
Bezeichnung	Schuh-Foto
Beschreibung	Zum Steckbrief des Schuhs soll ein Foto erfasst werden können
Vorbedingung	Req-0004
Zusatz-Info	-
Status	offen

3. Umsetzung

Dieses Kapitel enthält konzeptionelle Informationen und Angaben zur Umsetzung.
Nach einer kurzen Analyse habe ich mich für die Umsetzung mit PHP entschieden. Dies, weil XQuery unterstützt wird, anderseits weil die Anforderung Req-0002 nach einer Weblösung verlangt.

3.1. Server

In diesem Kapitel sind Informationen zum serverseitigen Teil aufgeführt

3.1.1. XML-Datenbank

Damit die XML-Datenbank verwendet werden kann, sind folgende zwei Schritte nötig:

1. XML-Datenbankserver installieren
2. XML-Datenbank einrichten

Der erste Schritt inkl. minimalen Hintergrundinformationen zu nativen XML-Datenbanksystemen ist in Kapitel 4 ausführlicher dokumentiert

Der zweite Schritt gestaltet sich sehr einfach. Im *BaseX GUI* kann über den Menupunkt *Database-New* die folgende Maske (Abbildung 3.2) aufgerufen werden. Im rot markierten Bereich wird der Name der Datenbank festgelegt. Für diese Arbeit wird der Name *shoes* verwendet:

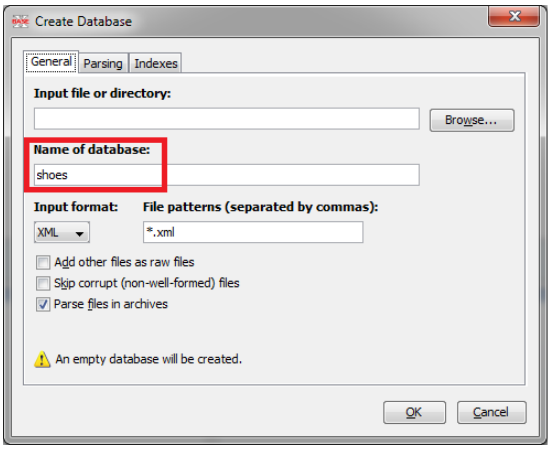

Abbildung 3.1.: BaseX - neue Datenbank

3.1.2. Aufbau XML-Struktur

Der Aufbau für die Steckbriefe ist relativ einfach aufgebaut. Das *shoe*-Element ist als Parent-Element definiert, alle Attribute sind entsprechend als Child-Element angehängt.

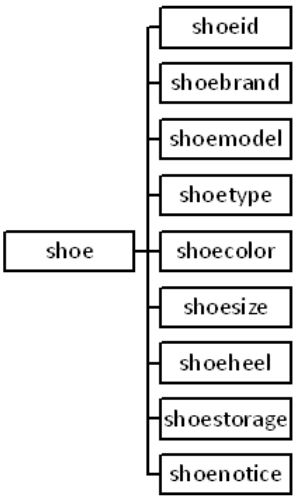

Abbildung 3.2.: XML-Struktur Schuh

Bereits beim Speichern des zu erfassenden Schuhs erfolgt die ersten Prüfungen auf Pflichteingaben (direkt durch das HTML-Formular). Konnte das Formular gespeichert werden, so wird aus diesen Angaben eine XML-Sequenz (= Steckbrief) generiert und in einer weiteren Prüfung gegen ein XML-Schema verglichen. Ist auch dieser Vorgang erfolgreich abgelaufen, wird das neue XML in der Datenbank angefügt.

Es stellt sich hier natürlich die Frage, ob die Daten in diesem Beispiel wirklich doppelt geprüft werden sollen. Diese Fragestellung wird an dieser Stelle jedoch nicht weiter verfolgt, es werden aber beiden Varianten verglichen.

Variante Eingabeformular

Der folgende Code-Auszug entspricht der Definition des Dropdown-Liste im Erfassungsformular für das Attribut Schuhfarbe:

```
<select name="shoecolor" ref="shoecolor" required="required" >
    <option value=""></option>
    <option value="Anthrazit">Anthrazit</option>
    <option value="Blau">Blau</option>
    <option value="Gelb">Gelb</option>
    <option value="Grau">Grau</option>
    <option value="Gruen">Gruen</option>
    <option value="Helblau">Helblau</option>
    <option value="Rosa">Rosa</option>
    <option value="Rot">Rot</option>
    <option value="Schwarz">Schwarz</option>
    <option value="Silber">Silber</option>
    <option value="Weiss">Weiss</option>
</select>
```

Die Eingabemöglichkeit wird auf Grund des Eingabefeldes mit einer fixen Auswahl bereits eingeschränkt. Es können so keine Fehleingaben erfolgen. Mit der Option *requiered='requiered'* wird zusätzlich geprüft, ob ein Wert ausgewählt wurde (wobei die erste Option mit Länge 0 nicht als Eingabe gilt).

Variante XSD-Validierung

Die Validierung mittels XSD-Schema ermöglicht ebenfalls auf bestimmte Typen oder eben auch eine Auswahl zu prüfen. Hinzu kommen hier jedoch noch weitere Prüfmöglichkeiten wie z.B. der Datentyp oder die Kardinalität. Zustätzlich steht die Möglichkeit offen, eigene Typen (Kombinationen aus bereits existierenden Typen mit bestimmten Zusatzattributen) zu definieren, was diese Technik sehr flexibe und mächtig macht.

Das folgende Beispiel stellt einen Auszug aus dem XSD dar. Der erste Abschnitt besteht aus einer Schema-Definition, im folgenden mittleren Abschnitt ist ein komplexer Typ definiert. Dieser symbolisiert den ganzen Schuh mit seinen Eigenschaften. Im dritten Teil ist ein sogenannt simpleTyp definiert, dieser stellt einen Datentyp dar welcher aus bestimmten Farben besteht.

```
<xs:schema version="1.0"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           elementFormDefault="qualified">

    <xs:element name="shoe" />
     <xs:complexType name="shoe">
          <xs:sequence>
               <xs:element name="shoebrand" type="sbrand" />
               <xs:element name="shoemodel" type="xs:string" />
               <xs:element name="shoetype" type="stype" />
               <xs:element name="shoecolor" type="scolor" />
               <xs:element name="shoesize" type="ssize" />
               <xs:element name="shoeheel" type="sheel" />
               <xs:element name="shoestorage" type="sstorage" />
               <xs:element name="shoenotice" type="xs:string" />
          </xs:sequence>
          <xs:attribute name="shoeid" type="xs:string"/>
     </xs:complexType>

     <xs:simpleType name="scolor">
          <xs:restriction base="xs:string">
               <xs:enumeration value="Anthrazit"/>
               <xs:enumeration value="Blau"/>
               <xs:enumeration value="Gelb"/>
               <xs:enumeration value="Grau"/>
               <xs:enumeration value="Gruen"/>
               <xs:enumeration value="Helblau"/>
               <xs:enumeration value="Rosa"/>
               <xs:enumeration value="Rot"/>
               <xs:enumeration value="Schwarz"/>
               <xs:enumeration value="Silber"/>
               <xs:enumeration value="Weiss"/>
          </xs:restriction>
     </xs:simpleType>
```

Zusammenfassung

Zusammengefasst kann festgehalten werden, dass eine doppelte Validierung in dieser Applikation kaum Sinn macht.
Sind die Daten lediglich über ein bestimmtes GUI zu erfassen, so kann die Validierung dort mittels den HTML-Funktionen durchaus genügen. Werden die Daten z.B. von Fremdsystemen mittels Webservice eingepflegt, so kann mittels Schema dafür gesorgt werden, dass die Daten trotzdem den Anforderungen entsprechen, auch wenn dort gegebenenfalls nur eine vereinfachte Validierung stattfindet.

3.2. Frontend

Dieser Abschnitt beschreibt den Teil, welcher vom Benutzer verwendet wird.
Der Aufbau wird in einzelne Webseiten aufgeteilt und im Sinne eines Prototyps ohne spezielle Formatierung umgesetzt.

3.2.1. Webserver

Die Applikation wird mittels PHP und HTML realisiert. Als Webserver wird XAMPP (Standardinstallation) verwendet. Das Setup und die Anleitungen sind auf der Homepage von XAMPP [1] verfügbar.

3.2.2. Erfassungsmaske

Bei der Erfassung (Req-0004) werden nach Möglichkeit zur Vereinfachung Dropdown-Listen eingebaut. So kann bequem z.B. die Schuhgrösse aus einer Auswahl gewählt werden. So werden auch proaktiv Eingabefehler (Schreibfehler) vermieden.
Hier wäre es für eine weitere Version hilfreich, wenn mittels eines Backends die Einträge in den Listen erweitert werden können.

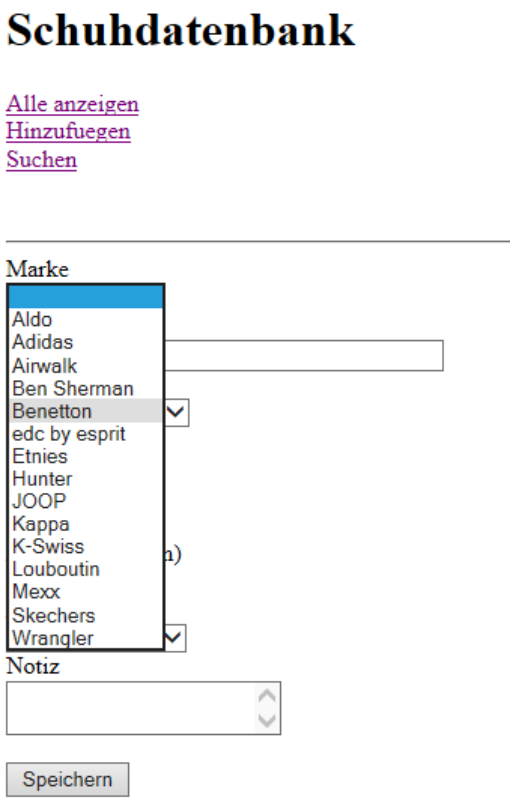

Abbildung 3.3.: BaseX - Schuh erfassen

3.2.3. Suchmaske

Damit der Benutzer bestimmte Daten suchen kann, muss die Datenbank auf geeignete Weise durchsucht werden können. BaseX unterstützt die Abfragesprache *XQuery*, mit welcher verschiedene Abfragen möglich sind. Folgend ein Auszug der *XQuery*-Syntax für die Suche:

Ausdruck	**Beschreibung**
/	ruft das Wurzelelement auf
//	ruft Knoten mit diesem Namen auf, egal wo sie stehen
.	ruft den eingegebenen Knoten auf
..	ruft das Elternelement des eingegebenen Knotens auf
@	ruft Attribute auf

Die Suche (Req-0006) soll die Suche über alle Steckbriefe ermöglichen. Mittels *XQuery* werden sämtliche Felder eines Steckbriefes (Ausnahme *Notiz*-Feld) abgefragt.
Die umgesetzte Suche ermöglicht die Eingabe eines Suchwortes. Das Programm prüft dann alle entsprechenden Felder auf allen Steckbriefen auf Vorhandensein dieses Suchwortes. Folgend die *XQuery*-Abfrage in der Schuh-Datenbank-Applikation:

```
xquery /shoe[(shoebrand='".$keyword."')  or
             (shoemodel='".$keyword . "')" or
             (shoetype="" . $keyword . "')" or
             (shoecolor="" . $keyword . "')" or
             (shoesize="" . $keyword . "')" or
             (shoeheel="" . $keyword . "')" or
             (shoestorage="" . $keyword . "')]
```

Das Elemente */shoe* definiert, dass sämtliche Schuh-Steckbriefe abgefragt werden (resp. zurückgegeben werden). Die Abfrage in den eckigen Klammern definieren das eigentliche Suchkriterium. Die angegebenen Elemente (z.B. *shoetype*) sind entsprechend Unterelemente (Child-) vom Schuh-Steckbrief.
Würde man die Suche kombiniert anbieten wollen (bestimmtes Schuhmodell in einer bestimmten Grösse), so müsste iterativ über sämtliche Steckbriefe mit jedem Suchwort eingegrenzt werden.

Auf Grund der zur Verfügung stehenden Zeit wurde hier nur die erste Variante umgesetzt:

Abbildung 3.4.: BaseX - Schuh Suche

3.2.4. Anzeigeliste

Für die Anzeigeliste aller erfassten Steckbriefe (Req-0005) kann ebenfalls auf eine *XQuery*-Abfrage zurückgegriffen werden. Um den kompletten Inhalt der Datenbank auszugeben, wird folgender Befehl verwendet:

```
xquery .
```

Das Ergebnis wird dann als HTML-Tabelle umformatiert und ausgegeben:

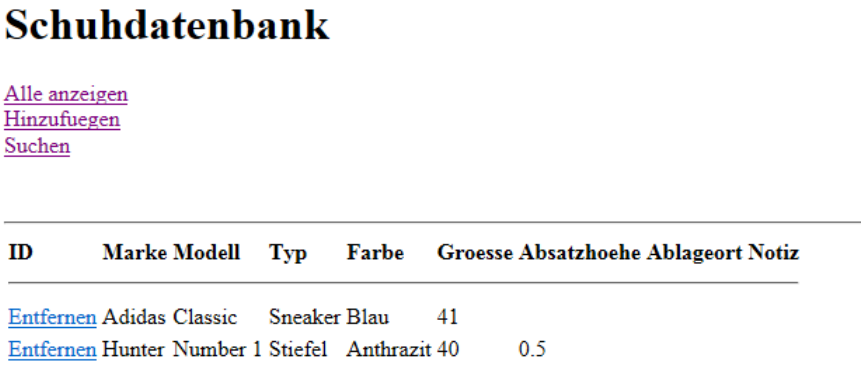

Abbildung 3.5.: BaseX - Schuh Anzeigeliste

3.2.5. Bild-Ablage

Gemäss der Anforderung Req-0009 soll im Steckbrief ein Bild abgelegt werden.
Die XML-Spezifikation sieht vor, das für diesen Fall ein JPEG-Bild mittels *base64_encode($Bild-Datei)* entsprechend codiert und im XML abgelegt wird. Die Umsetzung hätte den Zeitumfang der Arbeit noch weiter überstiegen, daher wurde hier nur der oben erwähnte Ansatz dokumentiert.

4. XML-Datenbanksystem

Gemäss der Anforderung Req-0001 muss für die Umsetzung ein natives XML-Datenbanksystem (XML-DBS) verwendet werden.

4.1. Einführung native XML-Datenbank

Die Funktionsweise einer nativen XML-Datenbank ist vergleichbar mit dem strukturierten Speichern von XML-Dokumenten in einer Verzeichnisstruktur auf der Festplatte.

Vorteile

- durch die baumartige Struktur der XML-Dateien können hierarchische Datenstrukturen sehr gut serialisiert werden
- Dateien können originalgetreu abgelegt (archiviert) werden
- schneller Zugriff auf komplette XML-Dokumente

Nachteile

- bei grossen Datenmengen nicht so performant wie ein relationales Datenbanksystem
- wegen Sperren auf Dokumenten-Ebene muss im Mehrbenutzerbetrieb mit höheren Wartezeiten gerechnet werden
- nicht indizierte Daten erfordern beim Zugriff ein Parsen der Daten
- jüngere Technologie, daher weniger ausgereift und weniger bekannt

Auf Grund der aufgeführten Nachteile kommen native XML-Datenbanksysteme nur noch selten vor. Bei der Recherche fand ich nur sehr wenige Applikation welche ein XML-DBS einsetzen (Auf Grund der Datum-Einträge unklar, ob das überhaupt noch eingesetzt wird). Praktisch alle heutigen relationalen Datenbanksysteme haben eine optimierte XML-Unterstützung und sind wesentlich performanter.

4.2. BaseX

Da ich keine Erfahrung im Bereich XML-DBS besitze, habe ich mich für das vorgeschlagene Produkt *BaseX* [2] entschieden.

4.2.1. Systemanforderungen

Gemäss *BaseX*-Homepage [2] wurden werden folgende Systeme getestet und somit unterstützt:

- Windows (2000, XP, Vista, 7)
- Max OS X (10.x)
- Linux(SuSE xxx, Debian, Ubuntu)
- OpenBSD (4.x)

Zusätzlich muss Java 1.6 (JRE) installiert sein.

4.2.2. Installation

Die Installation von *BaseX* wurde auf einem Windows 7 Rechner vorgenommen und gestaltete sich sehr einfach:

- passende Installationsdatei herunterladen [3]
- Setup-Datei starten und die Dialoge bestätigen (nach 6 Klicks ist BaseX am Standard-Pfad installiert)
- Icon auf dem Desktop doppelklicken um *BaseX* zu starten

4.2.3. Programmaufbau

Beim Start des *BaseX*-GUI erscheint folgende Maske: In diesem GUI können die adminis-

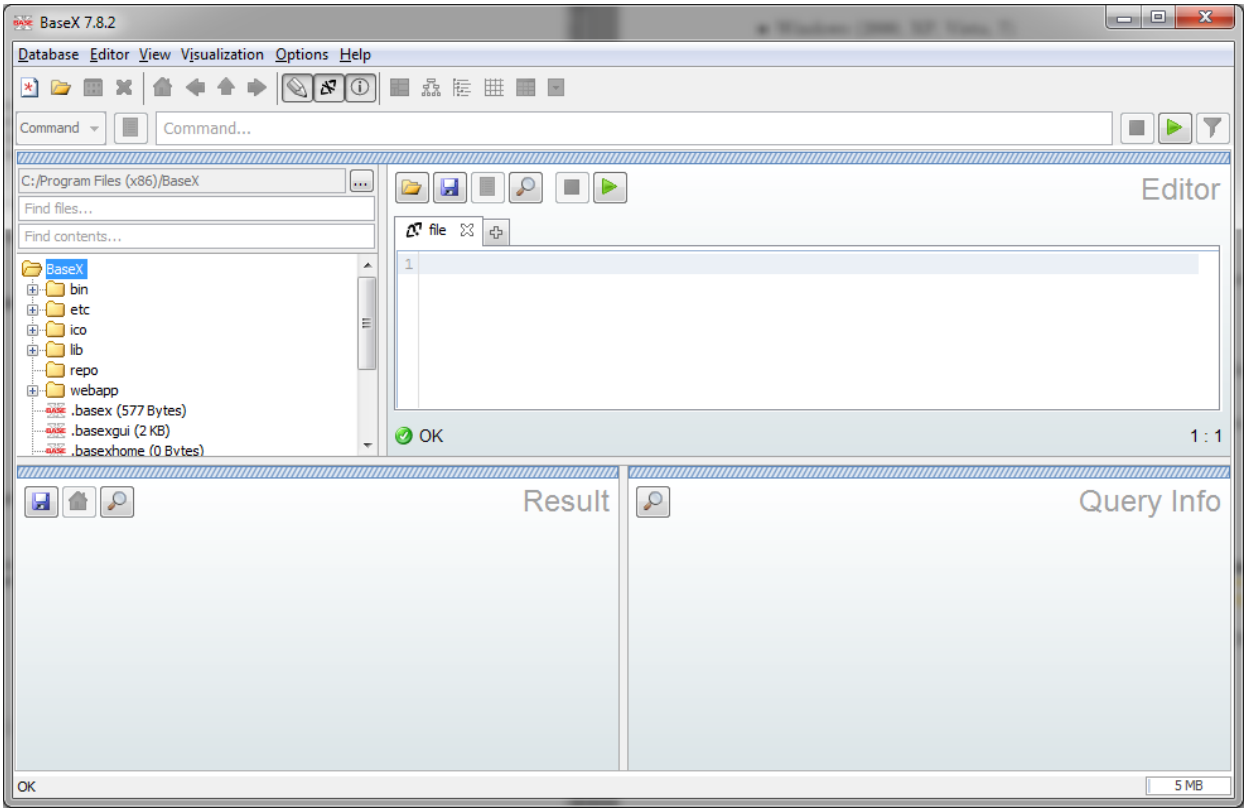

Abbildung 4.1.: BaseX-GUI

trativen Arbeiten durchgeführt werden. Auch lässt sich ein XML grafisch darstellen (Verschachtelte Quadrate symbolisieren die Verschachtelung im XML).

4.2.4. Server starten

Damit der Server im Netzwerk erreichbar ist, muss über das Menu *Database - Server-Administration* der Port definiert (Standard: 1984) und gestartet werden. Standardmässig lässt sich dann via Benutzer *admin* und Passwort *admin* auf die Datenbank zugreifen. Diese Angaben können entsprechend individuellen Bedürfnissen anpassen.

5. Fazit

Wer aus der relationalen Datenbankwelt einen Abstecher in die Welt der nativen XML-Datenbanken wagt, muss in einigen Punkten umdenken. Gerade die Abfragesprache *XQuery* ist im Vergleich zur *SQL*-Syntax bei neueren relationalen Datenbanksystemen wesentlich schwächer und umständliche (siehe Thema Suche).

Daneben muss vor allem auf den eingeschränkten Mehrbenutzer-Zugriff (resp. nur mit entsprechender Latenz wegen der Sperren) geachtet werden. Dieser wurde bereits beim Entwickeln als mühsam erachtet. Erfolgte der Zugriff aus der Applikation während die Datenbank im *BaseX-GUI* geöffnet war, so kam es zu einer entsprechenden Fehlermeldung.

Bezogen auf dieses Projekt lässt sich jedoch festhalten, dass hier ein natives XML-Datenbanksystem durchaus den Anforderungen entspricht.

Die Wahrscheinlichkeit jemals für ein Projekt ein natives XML-Datenbanksystem zu verwenden, erachte ich daher für mich als praktisch nicht vorhanden.

Abschliessend kann festgehalten werden, dass die entsprechenden Resultate der Aufgabenstellung erfolgreich ausgeführt werden konnten (mit Ausnahme der optionalen Anforderung Req-0009)

A. Glossar

Begriff	Beschreibung	Weblink
BaseX	**B**aseX Natives XML-Datenbanksystem	[2]
EBS	**E**lektronisches **B**ewertungs**s**ystem System der ZHAW	`ebs.zhaw.ch`
XAMPP	**K**omplett-Packet mit Apache Webserver und weiteren Komponenten	[1]
XML	E**x**tended **M**arkup **L**anguage	
ZHAW	**Z**ürcher **H**ochschule für **a**ngewandte **W**issenschaften	`www.zhaw.ch`

B. Abbildungsverzeichnis

(Einträge ohne Quellen-Angabe: eigene Darstellung)

C. Literaturverzeichnis

[1] Apache Friends. XAMPP. https://www.apachefriends.org, Stand: 08.06.2014.

[2] BaseX Team. BaseX. http://www.basex.org, Stand: 05.06.2014.

[3] BaseX Team. BaseX - Download. http://basex.org/products/download/all-downloads, Stand: 05.06.2014.

D. Quellcode

Auf das Anfügen von Quellcode wurde aus Platzgründen verzichtet. Den kompletten Quellcode findet man im GIT-Repository `https://github.com/MOtt/shoes`